Synthesizing, Editing, and Animating 3D Faces

by

Safa C. Medin

S.M., Massachusetts Institute of Technology (2021)
B.S., Boğaziçi University (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2025

Authored by:	Safa C. Medin Department of Electrical Engineering and Computer Science August 15, 2025
Certified by:	Gregory W. Wornell Sumitomo Professor of Engineering, Thesis Supervisor
Accepted by:	Leslie A. Kolodziejski Professor of Electrical Engineering and Computer Science Chair, Department Committee on Graduate Students

Synthesizing, Editing, and Animating 3D Faces

by

Safa C. Medin

Submitted to the Department of Electrical Engineering and Computer Science
on August 15, 2025 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

Recent advancements in computer vision and graphics have led to remarkable achievements in various visual computing tasks—from 3D reconstruction to real-time editing and animation of complex scenes. Within this landscape, modeling and manipulating human faces has emerged as a compelling topic, owing to its broad impact across multiple industries. This thesis addresses the inverse problem of estimating 3D realizations of human faces from 2D image observations, a formidable task that requires accurate modeling of the intricate geometry, appearance, and dynamics of faces. In the first part of the thesis, we focus on photorealistic manipulations of a single face image, where the manipulations are characterized by a 3D control input such as shape, albedo, and lighting. We design a framework to achieve a fast manipulation system that changes the attributes of a face image in a fully disentangled way. In the second part, we look into the problem of efficient rendering of dynamic 3D faces, where we develop a representation that is backward compatible with traditional graphics pipelines, does not require custom integration of machine learning tools at rendering time, and runs at interactive frame rates on consumer devices. Our representation leverages recent advances in neural rendering to sparsely sample the scene and model volumetric effects. In the final part of the thesis, we focus on the problem of animating volumetric 3D head avatars, where we develop a representation of controllable photorealistic avatars that can be rendered in real-time and have compatibility with existing hardware, rendering software, and modern network and streaming infrastructure to facilitate a faster adoption of virtual telepresence systems. To achieve this, we learn a layered mesh and blend textures that model appearance and deformations in the UV-space, and a simple linear transformation that maps tracked face model parameters to blend weights, enabling the deployment of platform-agnostic and streamable avatars to legacy renderers.

Thesis supervisor: Gregory W. Wornell
Title: Sumitomo Professor of Engineering

Acknowledgments

The path to a PhD is often described as a marathon, but for many students, it also involves frequent *sprints* that relentlessly test their endurance and resilience, all while they navigate the responsibilities and challenges of adult life. It would have been impossible for me to make it through this marathon alone, and I am deeply grateful to the many people who ran parts of it alongside me, some who stepped in at just the right moments, and others who stood by me *constantly* from the very beginning to the very end.

Without a doubt, the most influential *constant* throughout my PhD has been my advisor, Prof. Gregory W. Wornell. Since joining his group in 2019, his extensive knowledge and expertise, along with his unwavering support and encouragement, have helped me build confidence and find purpose in my path as a researcher. He taught me how to approach research problems from first principles, how to ask the right questions, and above all, how to place every problem within a broader intellectual context. He encouraged me, again and again, to zoom out, to reflect, and to consider the adjacent research questions that are just as important as the one in front of me. The research environment he has cultivated in our group is truly unique, one where each of us was given the freedom to explore a rich and diverse range of topics. This played a crucial role in helping me discover where my true interests lie in an intellectually fulfilling way. Thank you, Greg, for believing in me, for your guidance, and for your continued support.

Next, I would like to extend my heartfelt thanks to Prof. Bill Freeman and Prof. Frédo Durand for serving on my thesis committee, and for the invaluable feedback and guidance they have offered—not just during the final stages of this thesis, but throughout many moments in my journey, beginning with our collaboration during my very first project at MIT. I'm especially grateful to Frédo for the opportunity to serve as a teaching assistant for his computational photography course, which I immensely enjoyed. I would also like to thank Bill for organizing the computational imaging meetings for many years, which gave me the chance to hear about some of the most exciting research happening at MIT and beyond.

I was first introduced to the fields of 3D computer vision and graphics during my internships at Mitsubishi Electric Research Laboratories (MERL), marking a turning point in my career. I am deeply grateful to Dr. Tim Marks for giving me the opportunity to pursue a project outside my comfort zone and for his support and guidance throughout my time at MERL. Among the many collaborators I had the pleasure of working with there, I would especially like to thank Prof. Xiaoming Liu for offering valuable insights as I stepped into this new research area, and Prof. Bernhard Egger for his mentorship and patience as I learned to navigate this unfamiliar research realm. These early experiences ultimately inspired me to pursue a PhD focused on 3D vision and graphics, and my continued interaction with Bernhard beyond the internship has been both personally meaningful and intellectually rewarding, something I will always cherish.

During my time at MIT, I had the privilege of spending a significant part of my research journey interning at Google AR/VR, where I worked on the very exciting domain of 3D digital humans. There are no words that can fully express my gratitude to my long-time manager, Dr. Abhimitra Meka, who welcomed me to his team during my first internship in San Francisco and continued to support me through my subsequent internships at Google Cambridge. The projects he entrusted me with sparked a lasting interest that continues to shape my work, and his guidance was instrumental in my development as a researcher. I would like to thank Dr. Thabo Beeler for his support during my time at Google and for making my time on the Syntec team especially rewarding and enjoyable. I am also thankful to many colleagues at Google with whom I had the pleasure of collaborating. In particular, I would like to thank Dr. Ruofei Du for his invaluable assistance on my projects, as well as Dr. Yinda Zhang and Dr. Stephan Garbin for the insightful discussions that greatly contributed to my research progress.

I owe a special debt of gratitude to my former undergraduate research advisors, Prof. Vivek Goyal at Boston University and Prof. Bülent Sankur at Boğaziçi University. You were the first to introduce me to the world of research, guiding me through my earliest steps with patience, insight, and encouragement. Although our paths have taken different directions over the course of my PhD, the lessons I learned under your mentorship have quietly yet consistently accompanied me throughout my time at MIT, and I am deeply grateful for the foundation you laid.

Outside of my professional circle, I have been incredibly fortunate to have so many wonderful people around me, both in Cambridge and across the world, including my friends and those who have essentially become a family to me. I want to thank the past and present members of the Signals, Information, and Algorithms Laboratory, Tejas Jayashankar, Amir Weiss, Mumin Jin Sass, and Tricia O'Donnell, for their cherished presence, assistance, and the many fond memories we have shared. I want to express a special thank you to Armand Comas, Fatih Dinç, Doğa Doğan, Unay Dorken, Fazilet Gökbudak, Eren Kızıldağ, and Ozan Yakar for being there for me, no matter where in the world they happen to be. And lastly, I would like to express my utmost and most heartfelt gratitude to Ivan Laryionenka, who has been a *constant* source of happiness throughout most of my years at MIT. The memories we have created together and your endless encouragement and support have carried me through this long journey, for which I will forever be thankful.

Finally, I would like to thank my parents, Emine and Yalçın, and my sister, Merve, for being a *constant* source of love and compassion throughout my entire life. Your presence, whether near or far, has always brought me immense joy, and your support has always lifted me when I needed it most. You have undeniably played a profound role in shaping the person I have become today. I dedicate this thesis to you, and I feel incredibly lucky to have you as my family.

Contents

List of Figures

List of Tables

1 Introduction

Faces are perhaps our most expressive tool for human-to-human communication. A fleeting smile, a raised eyebrow, or a subtle glance can sometimes say more than paragraphs of text or minutes of speech. Unlike words alone, our faces carry layers of emotions, intentions, or cues that enrich our everyday interactions. This richness makes them not only essential to human connection but also a fascinating subject of study for researchers in the fields of computer vision, computer graphics, and human-computer interaction. Over the past few decades, breakthroughs in these fields have transformed numerous industries, including entertainment and media, gaming, security and surveillance, healthcare, retail, and fashion [1–3]. From digital actors that headline blockbuster films to artificial intelligence (AI) systems that help diagnose medical conditions by analyzing facial cues [4], advances in modeling and manipulating faces are quietly reshaping how we live, work, and communicate with each other.

Before we seek to develop algorithms for face-related applications, we must first ask a fundamental question: how can we represent a face, or more broadly, an object or a scene in a digital medium? For much of the digital era, the answer has been two-dimensional (2D) images consisting of arrays of pixels. The first digital image, illustrated in Figure 1.1, was created in 1957 by Russell Kirsch, who scanned an analog photograph of his infant son into a 176×176 pixel array [5]. Nowadays, billions of high-resolution images, such as the one displayed in Figure 1.2, are captured every day by devices we carry in our pockets. Digital cameras, once rare and expensive, are now ubiquitous, and so too are the images they produce—whether photographed by a human, painted digitally by an artist, or conjured entirely by machines. In fact, this thesis is written at a time when the distinction between *captured* reality and *generated* reality is increasingly blurred: many of the pixels we encounter on our screens no longer come directly from the physical world. This major shift raises exciting possibilities and, at the same time, complex challenges for how we process and interact with visual information.

Two-dimensional image representations, along with their dynamic counterparts *videos*, remain at the core of computer vision, powering mainstream applications such as classification, semantic segmentation, object detection, or scene understanding [6–8]. For many applications, however,

Figure 1.1: The first digital image was created by Russell Kirsch, who scanned a photo of his son, Walden.

Figure 1.2: A digital image taken by a mobile phone, showing Walden Pond State Reservation, in Concord, Massachusetts. Today, billions of images like this are captured every day using these devices.

2D representations of scenes are insufficient to model complex phenomena occurring in three-dimensional (3D) reality. For example, in the visual effects industry, the growing demand for photorealism requires synthesized scenes to be not only realistically animated but also to exhibit appearance changes that respond faithfully to dynamic scene conditions such as lighting variations. Achieving this level of fidelity naturally calls for 3D-grounded modeling of scenes. Indeed, in this thesis, we are motivated by various face-related applications that *demand* 3D modeling, and hence focus our attention on 1) how to synthesize 3D faces, 2) how to edit 3D faces with respect to various attributes, and 3) how to animate 3D faces.

Synthesizing 3D faces. Three-dimensional face synthesis refers to the process of generating digital representations of 3D human faces. This may be broadly categorized into two canonical problem settings: 1) creating new 3D faces by sampling from a learned distribution, and 2) reconstructing a faithful representation of a specific individual from their images captured under multiple viewpoints and/or varying scene conditions to produce a *digital twin* that mirrors the geometry, appearance, and dynamics of the individual. In the first scenario, *generative models* are often employed to capture the statistical variations of human face geometry and appearance with varying complexities. Earlier approaches describe these variations using linear subspaces [3, 9], enabling parametric control but producing only coarse, non-photorealistic results. Meanwhile, modern methods leverage high-capacity non-linear models using volumetric representations to synthesize more photorealistic faces, sometimes indistinguishable from real ones [10–12]. Importantly, these generative models are not merely artistic tools for creating synthetic faces. They often play a crucial role in tasks such as 3D face reconstruction, where statistical face priors can be integral to various solution algorithms [13–15]. The second scenario, *i.e.*, creating digital twins, has gained significant momentum with the rise of applications in extended reality (XR), an umbrella term we will use to refer to augmented reality (AR), virtual reality (VR), and mixed reality (MR) [16, 17]. Among these applications, virtual telepresence, an emerging technology that enables users to experience the sense of being physically present in a remote location by transmitting immersive audiovisual content in real time, serves as the culminating focus of this thesis. Our exploration spans algorithms for synthesizing 3D face representations tailored to the distinct application needs and data regimes of these technologies.

Editing 3D faces. For many face-related tasks, 3D face synthesis is often just the starting point. Certain applications require subsequent editing of various face attributes, such as 3D shape, appearance, head pose, or facial expression [18–20]. Achieving this task requires not only a physically accurate model of the face but also precise control systems that can adjust attributes in response to user input or algorithmic constraints. These systems often need to balance multiple competing requirements, such as high visual quality, computational efficiency, and disentanglement, which is a property that ensures variations in one attribute do not influence or *entangle* with other attributes. To meet these requirements, our key insight is to devise *3D-aware* editing approaches that are coupled with the 3D face synthesis process itself, hence requiring the joint modeling of facial geometry and appearance.

Animating 3D faces. Animating a 3D face involves generating temporally coherent sequences of deformations that reflect realistic face dynamics. Unlike one-off expression edits, face animation requires models that produce smooth transitions over time, capturing nuances like muscle motion, skin sliding, and subtle asymmetries [21]. Many animation algorithms require full controllability: these systems must respond to a set of input parameters (whether derived from motion capture, audio signals, or other modalities) to drive arbitrary talking sequences, facial expressions, or even idiosyncratic gestures. A key application motivating the development of such algorithms is virtual telepresence in XR, where participants equipped with head-mounted displays can interact in shared virtual environments. Facilitating these applications often involves a relay race that starts with tracking facial expressions using specialized hardware, using these as control inputs to drive an animation model, and rendering 3D faces on edge devices [22–24]. These pipelines typically need to address various considerations such as the photorealism of the output, rendering speed, and memory efficiency. To ensure the scalability of such pipelines, akin to today's 2D video communication systems, factors such as streamability and backward compatibility with existing hardware and software emerge as critical considerations. In this thesis, we focus on developing novel, dynamic 3D representations that address these practical constraints, which are often overlooked in modern approaches.

At the heart of these three challenges, *i.e.*, synthesis, editing, and animation, we position the *3D scene reconstruction* problem, which involves generating a 3D model of an object or scene from a set of observations, such as images or other sensor data [2, 25–27]. Although we exclusively focus on reconstructing faces, 3D reconstruction of *general scenes* has been studied for decades and underpins applications across an astonishingly wide range of fields. In robotics, reconstructed models of the environment enable autonomous agents to navigate, manipulate objects, and interact safely with their surroundings [28, 29]. 3D reconstruction is used to create immersive and realistic virtual experiences, such as in video games and simulations [30, 31]. It is also used in augmented reality applications to overlay digital information onto the real world [32]. Beyond these domains, recovering 3D objects from limited number of observations can be used in medical imaging to create detailed 3D models of the human body [33–35], allowing for more accurate diagnoses and treatment planning, or it can be used to create digital models of physical objects for design and prototyping purposes, as well as for quality control and inspection in manufacturing processes [36, 37]. 3D reconstruction methods can also be leveraged to create models of geological and archaeological sites or historical landmarks, allowing scientists to better understand and study these environments in a virtual setting, without the need for physical access [38, 39].

A growing number of applications of 3D scene reconstruction now operate on dynamic scenes, where the captured scene evolves over time as objects move, rotate, deform, or appear under varying lighting conditions [40–42]. Addressing such scenarios falls under the domain of *dynamic* 3D scene reconstruction, which introduces significant additional challenges. Beyond reconstruction, many applications also require the ability to control or animate the scene under novel conditions not present in the collected data. This elevates the reconstruction task into a *learning* problem. In this setting, the learning algorithm must be carefully designed to strike the right balance, offering sufficient flexibility to model scene variations while also providing enough regularization to avoid overfitting, as many different realizations of the reconstructed scene may potentially explain the same observed data. In this thesis, we approach the controllable dynamic face reconstruction problem through the lens of *inverse problem-solving*, providing a powerful foundation for guiding our algorithmic design and methodologies. This perspective builds on a long-standing history of inverse problems, which arise not only in the computer vision domain but also across many scientific and engineering disciplines [43–46].

While this thesis places the 3D face reconstruction problem at its core, each of the main chapters is shaped by a distinct task and application, which ultimately dictates the design of the methods we devise. At the forefront of these decisions lies the *3D face representation*, which profoundly influences the capabilities and limitations of any system we develop. Here, we critically examine the trade-offs between surface-based and volumetric representations, as well as between continuous and discrete formulations. These representations directly impact our key considerations such as photorealism, rendering efficiency, memory footprint, and other practical constraints. Throughout this thesis, we argue that the representation serves as the *secret sauce* for satisfying the diverse requirements of synthesizing, editing, and animating 3D faces. Yet, as we will see, representation is not the only consideration at play. Achieving practical and scalable solutions also demands attention to other factors, such as the computational speed of the developed algorithms or generalization capabilities to more challenging scenarios. These constraints, in turn, shape our choices of model architectures, training datasets, and loss functions. Together with the representations, these elements form a cohesive framework that advances the state of 3D-aware methods in face modeling and manipulation.

Thesis Outline

The work presented in this thesis is centered around the inverse problem of estimating 3D realizations of human faces from 2D image observations, a challenging task that requires accurate modeling of the intricate geometry, appearance, and dynamics of 3D faces. Beyond this reconstruction problem, we also explore how to enable fast, efficient, and photorealistic editing and animation of 3D faces, unlocking applications like portrait relighting, face image manipulation, and efficient rendering. Our ultimate goal is to synthesize and animate volumetric 3D face avatars suitable for virtual telepresence, an emerging technology that offers immersive alternatives to traditional video-based communications. In addition to photorealism and rendering efficiency considerations, a major focus of this thesis is ensuring that the developed 3D face representations are compatible with existing streaming systems, rendering platforms, and extended reality device ecosystems to facilitate a faster adoption of telepresence systems and making them a ubiquitous technology accessible to a wide range of users worldwide.

In Chapter 2, we present the foundational background for this thesis, with a focus on key topics of inverse problems in computer vision, 3D scene representations, and 3D scene reconstruction.

In Chapter 3, we explore the problem of rendering manipulated 2D realizations of a given single face image, where the manipulations are characterized by a control input that includes shape and albedo of a reconstructed 3D face mesh, as well as scene lighting conditions. Using only a dataset of single-view 2D face images, our objective in this chapter is to design a fast manipulation system that changes the attributes of a face image in a fully disentangled way, so that changing a single face attribute does not influence the other attributes.

In Chapter 4, we look into the problem of efficient rendering of dynamic 3D faces, where we aim to synthesize face models from a multiview video data of different subjects. Different from other chapters, we overlook the controllability aspect of the reconstructed model, and instead estimate a sequence of 3D geometry and appearance realizations. Our main goal in this chapter is to design a novel representation that is backward-compatible with traditional graphics pipelines without requiring custom integration of machine learning tools at rendering time.

In Chapter 5, we focus on the problem of animating volumetric 3D face avatars for applications in virtual reality and telepresence. Given multiview video data of a subject, we devise a framework to synthesize a controllable and photorealistic *digital twin* of this subject that can be rendered in real-time and have backward compatibility with existing hardware, rendering software, as well as modern network and streaming infrastructure.

In Chapter 6, we reflect on the presented work and highlight the key takeaways, situating them within the broader context of the current state of computer vision and artificial intelligence.

2 Background and Related Work

In this chapter, we review relevant foundational concepts in computer vision and graphics to contextualize the contributions of this thesis. We begin by formalizing the notion of inverse problems in vision, where we provide a statistical characterization of generic inverse problems and survey principal solution strategies, including optimization-based and learning-based approaches. Next, we introduce the two predominant classes of 3D scene representations—surface-based and volumetric—and detail key representations in these classes, including triangle meshes [47], signed distance functions [48], neural radiance fields [49], and 3D Gaussian Splatting [50]. Finally, we introduce the central problem in this thesis, the 3D face reconstruction task, and present a taxonomy of existing approaches categorized by their underlying 3D representations.

2.1 Inverse Problems in Computer Vision

Suppose we are given a system—whose characteristics may be partially or wholly unknown—that transforms an unobserved input into a collection of observable outputs. The problem of estimating the input or identifying the system characteristics from the observed outputs is known as an *inverse problem*, which arises in many fields in science and engineering, such as astronomy, biology, computational imaging, signal processing, as well as computer vision and graphics [43–46]. Solving inverse problems typically requires careful *system modeling* by creating abstract and often simplified representations of systems to analyze their structure and behavior more conveniently. These models can be statistical in nature, to account for prior information about the unobserved variables or randomness in the measurement processes. These assumptions lay the foundation for the choice and design of solution algorithms, which can involve a diverse class of approaches, such as closed-form inversions [51], classical iterative solvers [52], or deep neural network models [53].

Perspective projection, a conventional way to represent 3D objects on a 2D plane, is believed to have been formalized by Filippo Brunelleschi, a renowned Italian architect in the 15th century [54]. Later in the 17th century, another renowned Italian architect, Andrea Pozzo, wrote a foundational manual that systematically explains how to construct and analyze perspective illusions to infer underlying architectural plans [55]. This can be considered a very early instance of an approach to an inverse problem in the vision domain, long before the emergence of *computer* vision as a field. Nowadays, numerous visual computing tasks are formulated as inverse problems, including deblurring [56], denoising [57], super-resolution [58], demosaicing [59], as well as 3D tasks such as reconstruction [27], depth estimation [60], and inverse rendering [61]. Generally, these problems are defined through a *forward model* of the form

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{w}\,, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^m$ is the unobserved or *hidden* variable of interest, $\mathcal{A} : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ is the measurement operator that transforms the hidden variables through some process, $\mathbf{w} \in \mathbb{R}^n$ denotes an additive noise (*independent* of $\mathbf{x}$) due to uncertainities in the measurement process, and $\mathbf{y} \in \mathbb{R}^n$ denotes the observations collected from the system. Typically, inverse problems in computer vision cannot guarantee the existence or uniqueness of a solution, and hence are characterized as ill-posed problems [62]. This may require optimization-based techniques to settle for reasonable (but not necessarily correct) solutions when no solutions exist or regularizers to constrain the solution space when multiple (potentially infinitely many) solutions satisfy the observations [63].

2.1.1 Statistical Characterization of Inverse Problems

When an inverse problem has infinitely many solutions, on what principle should we prune the solution space? What makes some solutions more *likely* than the others? These questions call for a statistical modeling of the problem, and in particular, describing how the observed data and the hidden variables are jointly distributed, which gives the full statistical characterization of the problem. Suppose that such a joint distribution $p_{\mathbf{y},\mathbf{x}}$ exists and that it can be factorized into a *likelihood* term and a *prior* term as

$$p_{\mathbf{y},\mathbf{x}}(\mathbf{y},\mathbf{x}) = p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x})\, p_{\mathbf{x}}(\mathbf{x})\,, \tag{2.2}$$

which, to keep things general, can be continuous or discrete. Given an observation $\mathbf{y}$, we can describe our *complete* knowledge of $\mathbf{x}$ via the posterior distribution:

$$p_{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y}) = \frac{p_{\mathbf{y},\mathbf{x}}(\mathbf{y},\mathbf{x})}{p_{\mathbf{y}}(\mathbf{y})} \propto p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x})\, p_{\mathbf{x}}(\mathbf{x})\,, \tag{2.3}$$

where the last relation follows from the assumption that $p_{\mathbf{y}}(\mathbf{y})$ is constant and positive given $\mathbf{y}$. To estimate a hidden variable $\mathbf{x}$ given an observation $\mathbf{y}$, let's consider the following optimization problem with the *maximum a posteriori* objective:

$$\hat{\mathbf{x}}(\mathbf{y}) = \arg\max_{\mathbf{x}} p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x})\, p_{\mathbf{x}}(\mathbf{x}) = \arg\min_{\mathbf{x}} [-\log p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x}) - \log p_{\mathbf{x}}(\mathbf{x})]. \tag{2.4}$$

Based on Equation 2.1, the likelihood term can be rewritten as $p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x}) = p_{\mathbf{w}}(\mathbf{y} - \mathcal{A}(\mathbf{x}))$, where $p_{\mathbf{w}}$ denotes the noise distribution. This term is therefore governed by $p_{\mathbf{w}}$, and it typically manifests itself as the data-fidelity term. For example, suppose the noise is distributed as zero-mean, isotropic Gaussian, *i.e.*, $w \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Then, we have $-\log p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x}) = \|\mathbf{y} - \mathcal{A}(\mathbf{x})\|_2^2/2\sigma^2 + \textit{const.}$, which measures how well the current estimate $\mathbf{x}$ satisfies the observations in a squared loss sense. Despite its simplicity, such Gaussian idealization can be a good approximation of the physical reality when images are collected as raw measurements (without compression) in well-exposed environments with low noise levels [64, 65]. For common image restoration problems, while the likelihood term typically yields a simple closed-form expression, the prior term encodes complex statistical phenomena about the signals we aim to recover. Assuming it has a closed form, we can write

$$\hat{\mathbf{x}}(\mathbf{y}) = \arg\min_{\mathbf{x}} \phi(\mathbf{y} - \mathcal{A}(\mathbf{x})) + \lambda R(\mathbf{x}), \tag{2.5}$$

where $\phi(\cdot)$ is the data fidelity term, $R(\cdot)$ is the prior term, and $\lambda > 0$ is a scalar weight. When dealing with ill-posed inverse problems, leveraging these priors is crucial not only for statistical correctness but also for algorithmic viability, as they help produce solutions that obey physics or human perception via tractable optimization problems. Yet, crafting priors that promote these *natural* world statistics can be a highly non-trivial problem.

2.1.2 Approaches to Solving Inverse Problems

One of the most prominent philosophies that has received widespread attention in the context of problem solving is the *Occam's Razor* principle, which, among many explanations of a phenomenon, instructs *choosing the simplest one* [66]. Suppose we are interested in estimating images (consisting of arrays of pixel values) and that we rearrange them as n-dimensional vectors. In this form, *natural* images occupy a very small manifold in this potentially very high-dimensional space [67]. These images typically manifest scale-invariant statistical regularities, have dominant low-frequency structure, and contain contours and edges that delineate objects at multiple scales [68]. Such a structure may be represented sparsely (*i.e.*, with signals having few non-zero coefficients) in the pixel domain, or transform domains via linear transformations such as the Fourier transform, wavelet transforms [69], curvelet transforms [70], or discrete gradient operations [71]. These ideas can be extended to 3D representations to promote, *e.g.*, more *natural* geometry or appearance estimations [72]. Historically, while these *hand-crafted* priors have led to notable achievements in many inverse problems in vision and are still used in conjunction with more complex algorithms, data-driven methods have quickly gained popularity owing to their flexibility and the rapid advancements in deep learning over the last decade [62, 73, 74].

Many data-driven methods learn a parametric function $f_\theta : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ that *inverts* a given observation $\mathbf{y} \in \mathbb{R}^n$ to an estimate of $\mathbf{x} \in \mathbb{R}^m$. Given a dataset of ground-truth pairs $\{(\mathbf{y}_i, \mathbf{x}_i)\}_{i=1}^N$ where N is large, deep learning methods have demonstrated remarkable performance over conventional techniques using neural networks parameterized by a large number of learnable parameters [53, 75]. Optimization of these networks are typically performed via stochastic gradient descent [76] using random mini-batches from the dataset, to minimize some distance between each of $\mathbf{x}_i$ and $f_\theta(\mathbf{y}_i)$, as well as some explicit regularization term for the network weights [77, 78]. Overall, this mirrors the optimization problem in Equation 2.5, with an additional (implicit) regularization incorporated in the network architecture, consisting of specific types of connections or activation functions. Such *inductive biases* [79] introduced by the networks can also be exploited in a single-sample optimization setting, through frameworks such as the Deep Image Prior [80]. As an alternative to one-shot solvers that require a single forward pass through a network, the technique of *algorithm unrolling* [81] expresses iterations of classical optimization algorithms as trainable layers, allowing the incorporation of physical knowledge (*e.g.*, if the forward operator $\mathcal{A}$ is known) into the estimation process.

Many data-driven approaches in the vision domain rely on paired examples of observations and hidden variables, which can be challenging to acquire for certain tasks. As an example, let's consider the 3D reconstruction problem, where $\mathbf{x}$ describes a *3D representation* of a scene, $\mathcal{A}$ denotes a projection operation, and $\mathbf{y}$ is an image observation of the scene. Suppose we are given a single observation of a 3D scene, as opposed to multiple observations from different angles. The reconstruction task in this case can become severely ill-posed and hence require good priors to estimate reasonable 3D realizations. If we are given a large dataset of single-view images, and if the projection operation is known or can be estimated reasonably well, we can leverage a data-driven approach as before. For example, we can learn a function f_θ that takes in an observation $\mathbf{y}$ and outputs an estimate of $\mathbf{x}$, *i.e.*, $\hat{\mathbf{x}} = f_\theta(\mathbf{y})$. These estimations can then be cascaded with the forward operator and be supervised to match the observations by minimizing some distance between $\mathcal{A}(f_\theta(\mathbf{y}))$ and $\mathbf{y}$ [82]. Alternatively, one can leverage a *generative* approach by learning a function G_θ that transforms samples $\mathbf{z}$ from a known distribution to samples from the underlying distribution of the 3D scenes $p_{\mathbf{x}}$, *i.e.,* $\mathbf{x} = G_\theta(\mathbf{z})$. Then, we can subsequently embed a given observation $\mathbf{y}$ into the latent space of this generator so that $\mathbf{y} \approx \mathcal{A}(G_\theta(\mathbf{z}))$ [10, 14]. While such approaches traditionally focus on a specific class of scenes—such as faces [83, 84], animals [85], cars [86], or bedrooms [87]—more recent generative frameworks such as diffusion models [88–90] have led to higher capacity models trained with larger datasets, unlocking new paradigms for many inverse problems in addition to 3D reconstruction [91].

The recent advancements in *generative AI*, pioneered by diffusion models, have transformed the landscape of inverse problems in vision [91]. The unprecedented quality of complex, high-dimensional samples from these models has provided compelling evidence that these models can capture distributions that closely approximate the underlying data distribution, unlocking a wide range of algorithms with remarkable performance [92–94]. If the hidden variable $\mathbf{x}$ of an inverse problem is a natural image, a diffusion model trained on a large dataset of images can be utilized as priors for $p_{\mathbf{x}}$ by providing *smoothed* score functions $\nabla_{\mathbf{x}_t} \log p_{\mathbf{x}_t}(\mathbf{x}_t)$ of noisy data $\mathbf{x}_t = \mathbf{x} + \sigma_t \mathbf{z}$ at various noise levels t, with $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Using pairs of examples $\{(\mathbf{y}_i, \mathbf{x}_i)\}_{i=1}^N$, it is also possible to directly capture the posterior distribution $p_{\mathbf{x}|\mathbf{y}}$ through a conditional modeling [95, 96], which can allow sampling different estimations of $\mathbf{x}$ from a single observation $\mathbf{y}$. These probabilistic frameworks can be useful in applications such as medical imaging, where multiple explanations of the same data can be evaluated by humans to make more informed diagnoses [97].

When it is infeasible to collect a large amount of data for $\mathbf{x}$, as in the case of 3D reconstruction, diffusion models trained on a dataset of observations can still be used to solve inverse problems if $\mathcal{A}$ is known. Among these works, score distillation sampling [98] introduces a framework to generate 3D samples whose 2D projections are distributed according to the distribution captured by an image diffusion model. For the 3D reconstruction problem, we can interpret this framework as a general-purpose 3D prior that can be incorporated into gradient-based optimization algorithms by adjusting the gradients for the 3D estimations. Given a dataset of 2D observations only, Diffusion with Forward Models [99] modifies the reverse diffusion process by first lifting the 2D observations to 3D and subsequently rendering them from the given views, thereby explicitly modeling the distribution of 3D realizations given 2D observations. Although this method requires at least two observations of the same scene for training, a single image can be run through the model at inference time, producing realistic and diverse 3D realizations of the image.

While our discussion so far has alluded to methods that estimate 3D scenes, it has been agnostic to how 3D scenes are *represented*, which directly influences the design and development of algorithms for various tasks in the visual computing domain. In fact, a significant portion of this thesis is devoted to developing novel 3D representations of dynamic faces to meet the desiderata of applications in virtual reality and telepresence. In the next section, we look into various 3D representations that describe static or dynamic, general scenes.

2.2 3D Representation of Scenes

Everyday digital images we encounter or interact with are represented as arrays of pixels, which typically hold 3-channel, 8-bit color values. Such a representation is a direct result of how we conventionally *sense* images with consumer-level camera sensors that encode light intensities in red, green, and blue using color filter arrays [100]. Similarly, digital representations of 3D scenes can be imposed by the sensing technology, *e.g.*, point clouds may emerge as raw output data of the lidar technology [101], while voxel grids may arise naturally from volumetric medical imaging techniques such as computed tomography (CT) [34] or magnetic resonance imaging (MRI) [33]. Yet, depending on the application, these raw representations may prove to be suboptimal in terms of representational capacity or memory efficiency. For example, point clouds may be better utilized when converted into meshes using surface reconstruction algorithms [102, 103], or cubic memory growth of voxel grids can be prohibitive for high-resolution representations of sparse scenes, demanding alternative volumetric representations [104, 105].

In computer graphics, a 3D scene or object is typically described by its *geometry* and its *appearance*. The geometry defines how the scene occupies the 3D space and refers to its shape or structure, while its appearance defines how it visually looks or interacts with lighting. In this section, we review both surface-based and volumetric representations, and explore how we can formally describe their geometry and appearance.

2.2.1 Surface Representations

To study surface-based representations, we first look into continuous formulations, where a scene is modeled as a continuous function over 3D space, such as signed distance functions or implicit fields [106]. We then transition to discrete representations, with a primary focus on triangle meshes, which serve as the *de facto* standard in modern graphics pipelines due to their widespread adoption, efficient rendering, and compatibility with existing hardware and software ecosystems.

2.2.1.1 Continuous Surface Representations

Suppose we seek to represent a real-world 3D surface on a computer, and that we *sense* this surface by probing it at multiple points in space, gathering all samples as a point cloud $\{\mathbf{x_i}\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^3,\ i = 1, 2, \ldots, N$. One way to describe a continuous surface $\mathcal{S}$ that passes through these samples is through a zero-level set of a scalar function $f : \mathbb{R}^3 \to \mathbb{R}$ that satisfies $f(\mathbf{x}_i) = 0$ for $i = 1, 2, \ldots, N$. This function *implicitly* defines the surface $\mathcal{S}$ as

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = 0\}. \tag{2.6}$$

It is evident that there exist infinitely many functions f that fall into this characterization for finite N, which demands constraining the solution space by promoting, *e.g.*, more regular, smoother surfaces resembling real-life objects. A classical approach to ensure smoothness is to approximate 3D surfaces as linear combinations of smooth, analytical functions as

$$f(\mathbf{x}) = \sum_i w_i \, \phi_i(\mathbf{x}). \tag{2.7}$$

Among many function families, polynomials, splines, wavelets, and radial basis functions have been historically very useful in representing 3D shapes with varying degrees of complexity [107–109]. A different class of algorithms solves differential equations to find an implicit surface that best approximates a point cloud, such as Poisson surface reconstruction (PSR) [103], which stands out as one of the most popular methods. Given an oriented point cloud (where each point has a surface normal attribute that can be estimated using neighboring points), PSR solves the Poisson equation $\nabla \cdot \nabla f = \nabla \cdot \vec{V}$, where $\vec{V}$ is a vector field defined by smoothed, weighted aggregation of the input point normals. Recently, neural network-based parameterizations of surfaces offer more expressivity for representing complex surfaces and better compression capabilities as they can store entire scenes within small MLPs [106, 110], while still allowing fast rendering thanks to modern GPU hardware. Yet, classical methods like PSR remain as fast and predictable approaches to surface reconstruction when dealing with sufficiently dense point cloud data.

Among numerous classes of functions whose zero-level sets define a surface, signed distance functions (SDFs) [48] have been central in many applications that deal with continuous boundaries of (closed) 3D objects. Let Ω denote the set of points occupied by a 3D object and $\mathcal{S} := \partial\Omega$ denote its boundary, both defined in the Euclidean space. We can define a signed distance function

$$f(\mathbf{x}) = \begin{cases} -d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \in \mathbf{\Omega} \\ d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \notin \mathbf{\Omega} \end{cases} \tag{2.8}$$

where $d(\mathbf{x}, \mathcal{S}) := \inf_{\mathbf{x}' \in \mathcal{S}} \|\mathbf{x} - \mathbf{x}'\|$ is the Euclidean distance of a point $\mathbf{x}$ to the set $\mathcal{S}$. Such a function satisfies the Eikonal equation $\|\nabla f(\mathbf{x})\| = 1$ almost everywhere, describing a unique regularity condition among all functions that share the same zero-level set [111]. When optimizing for an SDF, imposing this property can promote numerical stability [112]. Nowadays, modern solutions parameterize SDFs using MLPs or transformers, achieving remarkable quality in surface reconstruction not only from point cloud data, but also from multiview images [113–115].

Fully analytical or neural representations of 3D surfaces can compress complex scenes into a modest number of parameters, hence allowing very efficient storage of these scenes in memory. At the same time, rendering such compact representations may require expensive operations (such as forward passes through a neural network per pixel) to *decompress* the representation, which reduces the rendering efficiency. To address this, we can, for example, discretize the 3D space into a grid and pre-compute function values at each element in this grid, thereby trading off memory (space) for compute (time). This inherent memory–compute trade-off is central to the design of 3D scene representations, with research efforts focusing on developing novel representations that have more promising memory–compute profiles. This motivates our discussion of discrete surface representations, which constitute a cornerstone of this thesis.

2.2.1.2 Discrete Surface Representations

Although continuous surface representations such as SDFs are mainstream tools in many computer graphics applications, piecewise linear discretizations of these surfaces using polygonal meshes have significantly more prevalence [47]. Because *triangle* meshes dominate real-time rendering in practice, let's focus our attention on formally defining a triangle mesh. Given a continuous 3D surface $\mathcal{S}$, suppose we *tessellate* it into triangle *primitives* at a specific resolution, effectively approximating the surface function with a piecewise linear one. We can then describe a single triangle mesh $\mathcal{M} := (\mathcal{V}, \mathcal{F}, \mathcal{A})$ as collection of three sets:

$$\mathcal{V} := \{\mathbf{v}_i \in \mathbb{R}^3 \mid i = 1, \dots, n_V\} \tag{2.9}$$

$$\mathcal{F} := \{\mathbf{f}_j = (i_{j1}, i_{j2}, i_{j3}) \in \{1, \dots, n_V\}^3 \mid j = 1, \dots, n_F\} \tag{2.10}$$

$$\mathcal{A} := \{\mathbf{a}_i = (a_{i1}, \dots, a_{in_a}) \in \mathbb{R}^{n_a} \mid i = 1, \dots, n_V\}. \tag{2.11}$$

Here, $\mathcal{V}$ is the set of vertex coordinates in 3D, which define the (shared) vertices of the triangles, where $n_V = |\mathcal{V}|$ is the number of vertices. $\mathcal{F}$ is the set of *faces*, where each ordered triple of indices $\mathbf{f}_j$ designates a (counter-clockwise) triangle with vertices $(\mathbf{v}_{i_{j1}}, \mathbf{v}_{i_{j2}}, \mathbf{v}_{i_{j3}})$, and $n_F = |\mathcal{F}|$ is the number of faces. Lastly, $\mathcal{A}$ is the set of vertex attributes, where n_a is the number of attributes per vertex. These attributes can, for instance, include colors for appearance, normals for shading, or 2D UV-space coordinates for texture mapping. These maps can be defined through a function

$$\mathcal{T} : [-1, 1]^2 \to \mathbb{R}^{n_a} \tag{2.12}$$

which takes a UV-coordinate $\mathbf{u} \in [-1, 1]^2$ and outputs an attribute vector $\mathbf{a} \in \mathbb{R}^{n_a}$. In particular, the *rendered* attribute of a surface point lying in face $\mathbf{f}_j$ is obtained by barycentric interpolation of its vertices' UV coordinates, followed by an evaluation of $\mathcal{T}$. For appearance modeling, these maps are typically defined in pixel space at a specific texture resolution, and can facilitate high-frequency appearance rendering even when the mesh resolution is not sufficiently high.

Triangle meshes are extremely popular in many production pipelines, as they are easy to create and share, can be edited with simple workflows to produce different assets and perform animations, can be rendered very efficiently with modern GPU hardware through rasterization, and virtually all graphics hardware and software support them natively [116]. For decades, meshes have been used to represent, edit, and animate 3D faces [117–120], spearheading the rapid progress that has been further accelerated by the 3D morphable face models [9]. At the same time, as the 3D face applications in the modern era demand photorealism to the point where renders are indistinguishable from reality, single-surface face meshes have been replaced in favor of *volumetric representations* that can handle complex geometry and appearance of hair, beard, eyes, or idiosyncratic details on skin [49, 50].

2.2.2 Volumetric Representations

Perhaps the most *natural* way to represent a 3D scene volume is to divide it into a uniform grid and assign attributes (such as color or opacity) to each cube in this grid [104, 121]. These *voxel*-based representations are nowadays a standard in many medical imaging applications as they naturally arise from the sensing process [33, 34]. For general scenes, while voxel-based representations have been historically used in many applications [105, 122–124], their memory profile is often prohibitive for high-resolution 3D scene reconstruction. Advances in deep learning have attempted to circumvent this problem by introducing low-resolution *deep* voxel grids that store high-dimensional features, which are mapped to color values upon projection via learnable functions [125–129]. While the idea of representing 3D scenes as continuous fields had previously been explored [130], the seminal work Neural Radiance Fields (NeRFs) [49] proliferated the use of MLPs for photorealistic and view-consistent rendering of high-resolution 3D scenes, effectively commencing the era of radiance fields in computer vision and graphics.

2.2.2.1 Neural Radiance Fields

Representation. To synthesize photorealistic views of complex scenes, it is essential to model how light interacts with matter along a ray. Neural radiance fields build on the classical volume rendering equation [131] and capture how light is emitted *and* absorbed at each point in space. Formally, suppose we are given a (virtual) camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o} \in \mathbb{R}^3$ is the camera origin, $\mathbf{d} \in \mathbb{R}^3$ is the (unit norm) camera ray direction, and $t \in \mathbb{R}$ is a scalar that parameterizes the ray progression from t_n to t_f, denoting near and far clips of the camera. Using an emission–absorption model, we can write the radiance received by the camera along ray $\mathbf{r}$ as

$$L(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(t)\, c(t)\, dt. \tag{2.13}$$

Here, $c(t)$ is the emitted radiance at position $\mathbf{r}(t)$, which typically holds color values in RGB. The volume density $\sigma(t)$ is the rate at which radiance is absorbed per unit length at $\mathbf{r}(t)$, while the transmittance $T(t)$ is the fraction of light that makes it from $\mathbf{r}(t_n)$ to $\mathbf{r}(t)$ without being absorbed. Using this interpretation, we can write the following relations [132]:

$$T(t + dt) = T(t)\,(1 - \sigma(t)dt) \tag{2.14}$$

$$\frac{T(t + dt) - T(t)}{dt} = \frac{dT(t)}{dt} = -T(t)\,\sigma(t). \tag{2.15}$$

Assuming a transmittance of 1 at the start of the ray (*i.e.*, $T(t_n) = 1$), the solution to the first-order ordinary differential equation in (2.15) can be written as

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(t)\, dt\right). \tag{2.16}$$

This indicates that a 3D scene (under the emission-absorption model) can be fully described by its volume density and radiance, which can be thought of as its geometry and appearance, respectively. The NeRF framework precisely does this by parameterizing a radiance field and a density field as a small MLP. To capture the view-dependent effects, the radiance field also depends on the ray direction $\mathbf{d}$. Formally, the full representation can be characterized by the weights of a *single* network $F_\Theta : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$, where $\mathbf{x} \in \mathbb{R}^3$ is a 3D position in the volume, and $\mathbf{c} \in \mathbb{R}^3$ is the color in RGB. In practice, computing the volume rendering equation in (2.13) requires approximating the integral via quadrature [133]. Suppose we are given N samples across a single ray at $t_1, \ldots, t_N$ that yield color and density values $\{(\mathbf{c}_i, \sigma_i)\}_{i=1}^N$ after querying the MLP. Assuming constant volume density along each interval $[t_i, t_{i+1})$, we can approximate the rendered color as

$$\hat{\mathbf{c}} = \sum_{i=1}^{N} T_i\,(1 - \exp(-\sigma_i\,\delta_i))\,\mathbf{c}_i, \text{ where } T_i := \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \tag{2.17}$$

where $\delta_i := t_{i+1} - t_i$ is the distance between consecutive samples. Defining $\alpha_i := 1 - \exp(-\sigma_i \delta_i)$ yields the traditional alpha-compositing equation

$$\hat{\mathbf{c}} = \sum_{i=1}^{N} w_i \mathbf{c}_i, \text{ where } w_i := \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{2.18}$$

If we conventionally let $\delta_N \to \infty$, we have $\alpha_N \to 1$ and $\sum_{i=1}^{N} w_i = 1$. In this case, the weights $\{w_i\}_{i=1}^{N}$ can be interpreted as probability masses that quantify the probability of a ray terminating at interval i after passing through previous intervals $1, \ldots, i-1$. This provides more stable gradients when optimizing a NeRF-based representation, which we focus on next.

Optimization. Suppose we are interested in reconstructing a volumetric representation of a static 3D scene using multiple observations of it from different angles. The NeRF framework [49] introduces 1) positional encoding to capture high-frequency components of the scene and 2) a hierarchical sampling strategy to more efficiently sample from the high-frequency representation. Specifically, positional encoding involves mapping the input of a function to a higher-dimensional space through an encoding function such as

$$\gamma(p) = \left(\sin(2^0 \pi p), \cos(2^0 \pi p), \ldots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)\right), \tag{2.19}$$

which uses L frequencies to map a single scalar in $\mathbb{R}$ to $\mathbb{R}^{2L}$. In typical NeRF-based models, this function is individually applied to each coordinate in the input positions and viewing directions. Meanwhile, the hierarchical sampling involves constructing two networks (a coarse one and a fine one) that implement an importance sampling procedure to gather samples from more informative regions of the scene, where the importance is derived from the weights w_i of each sample. To train these two networks, one can minimize a loss function that measures the pixel-wise discrepancy between the outputs of both networks and the ground truth training images, such as a squared loss used in the original work [49].

Variants. Neural radiance fields have revolutionized visual computing, leading to rapid advancements in many tasks, including reconstruction, view synthesis, relighting and appearance editing, scene understanding, and human digitization for AR/VR [134–136]. They have also been used to represent dynamic scenes using single- or multi-view videos of them, for applications such as free-viewpoint video synthesis, human performance capture, or content generation [40, 41, 137–139]. To represent motion in the scenes, one can modify the original NeRF formulation as $F_\Theta : (\mathbf{x}, \mathbf{d}, t) \mapsto (\mathbf{c}, \sigma)$, where t denotes the time-conditioning [140]. Alternatively, one can accompany a static MLP with a time-conditioned deformation field in 3D space to represent motion. We defer a more in-depth discussion of these methods to Chapter 4, where we develop a dynamic model based on the radiance field formulation.

Perhaps the most notable limitation of the original NeRF methodology [49] is that it represents an entire 3D scene with a global MLP, which is queried at *each* point along *each* ray, limiting the rendering speed and thereby the training speed. Among several works that attempt to circumvent this issue, Instant Neural Graphics Primitives (InstantNGP) [141] discards the heavy sinusoidal positional encoding and relatively large MLP queries, and it instead maps spatial positions to learned feature vectors stored in multi-resolution hash tables and uses much smaller MLPs to decode these features. Meanwhile, Plenoxels [142] represents scenes using sparse voxel grids, where each voxel stores density and radiance. As the radiance is merely stored as coefficients for a simple basis of view-dependent functions [143], the representation is completely neural network-free (and hence is not a *neural* field). At a high level, both of these works secure compute gains and reduce training and rendering times significantly by storing spatial scene features explicitly, thereby increasing the overall memory cost.

The original NeRF framework and its variants introduced so far rely on per-pixel ray casting to render scenes, by sampling and integrating radiance along camera rays. Since this stochastic procedure typically requires many queries along each ray, NeRF-based methods, including faster variants such as InstantNGP [141] or Plenoxels [142], face scalability challenges for high-resolution or interactive applications. The key idea that has gained attention recently is to trade off even more memory efficiency for computation, by designing GPU-friendly volumetric *primitives* that can be *rasterized* efficiently with modern hardware. One notable representation, Pulsar [144], introduces spherical primitives that can be rendered very efficiently, but the rather simple, depth-weighted blending of primitives limits its ability to represent complex scenes with volumetric effects. More recently, the seminal work 3D Gaussian Splatting (3DGS) [50] has introduced anisotropic 3D Gaussians as volumetric primitives that can be rasterized and alpha-composited very efficiently, securing significant speed gains over the previous state-of-the-art while achieving comparable image quality. As 3DGS and its subsequent variants marked a turning point in vision and graphics, leading to an explosion of methods that demonstrate notable performance gains in many visual computing tasks, we devote a separate section to 3DGS.

2.2.2.2 3D Gaussian Splatting

Representation. In the landscape of 3D scene representations, point-based methods leverage point-based primitives, which, in their simplest form, are merely fixed-size points that are disconnected from each other [145]. Although such a representation enables very fast rendering, the rendered images may manifest irregularities or holes, limiting their applications that demand high visual quality [146]. One approach to resolving this issue is to represent each point as *splats* in object-space or screen-space, allowing each primitive to extend beyond a single pixel when rendered [147, 148]. 3DGS follows the same spirit by representing scenes as clouds of 3D Gaussian blobs in the object space [50]. Formally, a single 3D Gaussian is defined through

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \tag{2.20}$$

where $\boldsymbol{\mu} \in \mathbb{R}^3$ and $\Sigma \in \mathbb{R}^{3\times3}$ are respectively the mean and covariance of the Gaussian, and $\mathbf{x} \in \mathbb{R}^3$ is a position in the 3D world-space. To constrain the covariance matrices to be positive semi-definite, the original work [50] factorizes them into scaling and rotation matrices as $\Sigma = RSS^{\mathrm{T}}R^{\mathrm{T}}$, where $R \in \mathbb{R}^{3\times3}$ is the rotation matrix and $S \in \mathbb{R}^{3\times3}$ is the scaling matrix. Each Gaussian is also associated with transparency $\alpha \in [0,1]$ and view-dependent radiance $\mathbf{c} \in R^c$ represented as coefficients to a basis of view-dependent functions, namely, spherical harmonics [143]. These functions are solutions to Laplace's equation in spherical coordinates and have the form

$$Y_\ell^m(\theta,\phi) = N_\ell^m\, P_\ell^m\,(\cos\theta)\, e^{im\phi}\,, \tag{2.21}$$

where Y_ℓ^m is referred to as a spherical harmonics function of degree ℓ and order m, P_ℓ^m is an associated Legendre polynomial, and N_ℓ^m is a normalization factor [149]. Here, the degree ℓ takes non-negative integer values while the order m follows $-\ell \leq m \leq \ell$. Therefore, there exists $2\ell + 1$ functions for degree ℓ, and $(\ell + 1)^2$ functions in total up to and including degree ℓ. Both degree and order determine the complexity of the function through the number of oscillations around the sphere on which they are defined [150]. To represent view-directional radiance through spherical harmonics, we can take the linear combination of these functions with $(\ell + 1)^2$ coefficients for each color channel, and associate these coefficients with each Gaussian primitive.

One major benefit of representing scenes as a cloud of Gaussian primitives is that these primitives can be projected onto screen space efficiently through rasterization, hence avoiding ray marching used in neural radiance field rendering. Concretely, each 3D Gaussian in the object space can (approximately) be projected to the pixel space as a 2D Gaussian with 2D mean and covariance. If a single pixel falls under multiple Gaussians, we can first sort them based on their depth and alpha-composite them using Equation 2.18, where each color channel of the i-th Gaussian is expressed as a linear combination of the real part of the functions in Equation 2.21, while alphas associated with each Gaussian are modulated with

$$G_p(\mathbf{u}) = \exp\left(-\frac{1}{2}(\mathbf{u} - \boldsymbol{\mu}_p^{\mathrm{T}})\,\Sigma_p^{-1}\,(\mathbf{u} - \boldsymbol{\mu}_p)\right). \tag{2.22}$$

Here, $\boldsymbol{\mu}_p \in \mathbb{R}^2$ and $\Sigma_p \in \mathbb{R}^{2\times 2}$ are respectively the mean and covariance of the *projected* Gaussian, and $\mathbf{u} \in \mathbb{R}^2$ is a position in 2D pixel space. In summary, a 3DGS-based representation in its simplest form can be expressed as

$$\mathcal{G} = \{(\boldsymbol{\mu}_i, \Sigma_i, \mathbf{c}_i, \alpha_i)\}_{i=1}^{N}\ , \tag{2.23}$$

where N is the number of Gaussians. Here, the position, shape, and density of Gaussians can be considered as the geometry of the representation, whereas the spherical harmonics coefficients describe the appearance. We should also note that the rendering of 3DGS is mathematically similar to evaluating a radiance field that has been discretized into a set of Gaussian kernels, each of which describes density and radiance continuously in 3D space. Therefore, 3DGS can be considered a *radiance field* as well—as reflected in the title of the original paper [50]—but it is not a *neural* radiance field, as there is no neural network involved in the original representation.

Optimization. Once again, suppose we are interested in recovering a volumetric representation of a static 3D scene using multiple observations of it from different views. The original algorithm [50] starts by initializing a set of Gaussians located at points obtained from an off-the-shelf structure-from-motion (SfM) algorithm [151]. To train the positions, shapes, colors, and transparencies of each Gaussian, one can again minimize a loss function that measures the pixel-wise discrepancy between the rendered and ground-truth images. To handle the discrete nature of the representation, the original work [50] proposes an adaptive density control algorithm that dynamically prunes low-contributing Gaussians and spawns new ones in regions with high error. While such a heuristic may work well in practice, it may also require more careful parameter tuning or lead to unstable optimization compared to the training of NeRF-based methods [152]. Nevertheless, the significant gains secured in rendering and training speed make 3DGS a very appealing representation.

Variants. Since the introduction of 3D Gaussian Splatting, the vision and graphics communities have revisited many tasks formerly addressed using NeRF-based methodologies [153, 154]. To represent dynamic scenes, one can naively learn a sequence of attributes for each Gaussian in the cloud. Similar to prior work in NeRFs, one can also designate a canonical space and describe motion via a deformation field that modifies position and shape attributes of Gaussians [155]. Another class of methods attaches Gaussians to simpler representations such as meshes, and performs animations by rigging these representations [156].

Revisiting the inherent memory–compute trade-off in 3D representations, 3DGS prioritizes fast rendering at the expense of high memory usage, exceeding the memory requirements of the fastest neural representations, such as InstantNGP [141]. Although neural representations enjoy some model compression by design, primitive-based representations such as 3DGS may require more careful memory-handling for consumer-facing applications. Such demand has motivated various 3DGS compression methods [157], which use techniques such as pruning of redundant primitives [158] or vector/codebook quantization of per-splat attributes [159, 160]. Another class of methods focuses on efficient streaming of dynamic 3D Gaussians to reduce the bandwidth requirements without compromising real-time performance [161–163], which is essential for production-level real-world applications such as virtual telepresence.

Beyond conventional 3D Gaussians, new representations have pushed 3DGS towards better surface accuracy and material realism. Works such as 2D Gaussian Splatting [164] and Quadratic Gaussian Splatting [165] collapse 3D primitives into disks or surface patches, achieving more accurate surface modeling. For more realistic appearance modeling, normals, shading functions, or material properties may be attached to Gaussians as additional attributes [166–168]. Collectively, these innovations transform Gaussian splats from fast radiance fields into a relightable primitive family that interfaces cleanly with mesh pipelines and future compression standards.

2.3 3D Face Reconstruction

In this chapter, we initially discussed inverse problems encountered in computer vision, with an emphasis on the reconstruction of *general* 3D scenes. Having provided an overview of how these general scenes are represented, we now turn our attention to the specific class of scenes this thesis focuses on: 3D human faces. Among many problems that involve 3D faces, we first limit our attention to the 3D face reconstruction problem, which lies at the core of this thesis.

2.3.1 Problem Definition

We define *geometry model* and *appearance model* of a 3D representation as follows:

$$G : \mathbb{R}^p \to \mathbb{R}^g \qquad A : \mathbb{R}^p \to \mathbb{R}^a \,, \tag{2.24}$$

both of which take a vector of scene control parameters $\mathbf{p} \in R^p$ and output the geometry and appearance *realizations* of the scene. For example, a geometry realization of a single mesh surface would include vertex positions as well as the mesh topology that defines how these vertices are connected, while an appearance realization may include an RGB texture map in UV-space as well as the UV-coordinates assigned to each vertex. For a neural radiance field, the neural network weights for the density and color fields would capture the geometry and appearance realizations of the scene, respectively. The control parameters can define the position, orientation, deformation of the objects, or the lighting conditions in the scene, making the definition in (2.24) general for any controllable 3D scene representation.

Given a set of camera parameters $\mathbf{c} \in \mathbb{R}^c$ that defines camera intrinsic and extrinsics, we assume that we are given a deterministic function $\mathcal{R} : \mathbb{R}^g \times \mathbb{R}^a \times \mathbb{R}^c \to \mathbb{R}^n$ that *renders* a 3D scene (described by its geometry and appearance) to a 2D image as seen from the camera. Putting everything together, we write our observation model as

$$\mathbf{y} = \mathcal{R}(G(\mathbf{p}), A(\mathbf{p}); \mathbf{c}) + \mathbf{n}, \tag{2.25}$$

where $\mathbf{n} \in \mathbb{R}^n$ denotes an additive noise vector due to the measurement process, and $\mathbf{y} \in \mathbb{R}^n$ is the observed image of the scene from a camera with parameters $\mathbf{c}$. To perform 3D reconstruction of a face from its single or multiple observations, we can leverage traditional techniques commonly applied to general scenes, including shape-from-shading [169], photometric stereo [170], multi-view stereo [25], and structured-light methods using active illumination [171]. However, to enhance the performance of these algorithms, it is natural to exploit some domain-specific knowledge about 3D faces. Given that reconstructing 3D scenes from limited observations inherently constitutes an ill-posed inverse problem, statistical modeling of faces would be a compelling approach to incorporate such knowledge. Specifically, although a 3D face representation can be extremely high-dimensional (for instance, a face mesh at moderate resolutions may contain thousands of vertices), the intrinsic degrees of freedom governing facial geometry and albedo are well-captured within a much lower-dimensional subspace. This insight has motivated research into statistical modeling of 3D faces, culminating in the seminal work on 3D Morphable Face Models (3DMM) by Blanz and Vetter [9], which has significantly advanced the field of 3D face analysis.

2.3.2 3D Morphable Face Models

Consider the underlying distribution of human faces across the global population. While each person possesses a unique combination of face shape and skin tone, there exist strong shared structural patterns—for instance, most faces exhibit bilateral symmetry around the sagittal plane, and skin tones generally fall within a limited range of naturally occurring hues. Given a sufficiently large collection of 3D face scans or other digital representations, it may be feasible to model a distribution that captures the dominant modes of variation within this data. In the original work, Blanz and Vetter [9] use RGB laser scans of faces of 200 young adults, where dense correspondences between facial features are estimated via an optic flow-based algorithm. Given a total number of vertices V in each scan, this yields a face shape vector $\mathbf{s} \in \mathbb{R}^{3V}$ and texture vector $\mathbf{t} \in \mathbb{R}^{3V}$ for each sample in the dataset. Then, they model shape and texture distributions individually as multivariate Gaussians whose parameters are estimated from the data:

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{B_s}\boldsymbol{\alpha}_\mathbf{s} \qquad \mathbf{t} = \bar{\mathbf{t}} + \mathbf{B_t}\boldsymbol{\alpha}_\mathbf{t}, \tag{2.26}$$

where $\bar{\mathbf{s}} \in \mathbb{R}^{3V}$ and $\bar{\mathbf{t}} \in \mathbb{R}^{3V}$ are estimated mean vectors for shape and texture, $\mathbf{B_s} \in \mathbb{R}^{3V \times N_s}$ and $\mathbf{B_t} \in \mathbb{R}^{3V \times N_t}$ are basis matrices obtained via truncating the eigenbases of the covariance matrices for shape and texture with $N_s, N_t \ll 3V$, and both $\boldsymbol{\alpha}_\mathbf{s} \in \mathbb{R}^{N_s}$ and $\boldsymbol{\alpha}_\mathbf{t} \in \mathbb{R}^{N_t}$ are distributed standard normally. Albeit simple, such a distribution can serve as a powerful prior for analysis, synthesis, and reconstruction tasks involving human faces. Indeed, subsequent 3DMMs use larger 3D datasets, incorporating facial expressions and full heads into the models [172–174].

Full head models. Perhaps one of the most influential morphable face models nowadays is FLAME [174], which models a full head with $K = 4$ joints: neck, jaw, and two eyeballs. Under this model, a canonical (unposed) face can be written as follows:

$$\mathbf{s}(\boldsymbol{\beta}, \boldsymbol{\psi}) = \bar{\mathbf{s}} + \mathbf{B}_{\text{id}}\boldsymbol{\beta} + \mathbf{B}_{\text{expr}}\boldsymbol{\psi}\,, \tag{2.27}$$

where $\bar{\mathbf{s}} \in \mathbb{R}^{3V}$ is the mean face shape, $\mathbf{B}_{\text{id}} \in \mathbb{R}^{3V\times|\boldsymbol{\beta}|}$ and $\mathbf{B}_{\text{expr}} \in \mathbb{R}^{3V\times|\boldsymbol{\psi}|}$ are the *identity* and *expression* PCA bases, and $\boldsymbol{\beta} \in \mathbb{R}^{|\boldsymbol{\beta}|}$ and $\boldsymbol{\psi} \in \mathbb{R}^{|\boldsymbol{\psi}|}$ are identity and expression coefficients. These two *offsets* on top of the template mesh $\bar{\mathbf{s}}$ explain the shape variations due to identity of the subject and due to facial expressions, although achieving full disentanglement of these variations is a challenging task [175]. Now, suppose we are given rotation vectors for each joint and a global rotation vector in axis-angle representation ($\in \mathbb{R}^3$), all of which are concatenated into a single vector $\boldsymbol{\theta} \in \mathbb{R}^{3K+3}$. We can then write *pose offsets* ($\in \mathbb{R}^{3V}$) as

$$\Delta\mathbf{p}(\boldsymbol{\theta}) := \mathbf{B}_{\text{pose}}\left(R(\boldsymbol{\theta}) - R(\boldsymbol{\theta}_0)\right)\,, \tag{2.28}$$

where $R : \mathbb{R}^{3K+3} \longrightarrow \mathbb{R}^{9K}$ converts pose vectors of all *joints* to vectorized 3×3 rotation matrices and concatenates them, $\mathbf{B}_{\text{pose}} \in \mathbb{R}^{3V\times 9K}$ includes the pose blendshapes, and $\boldsymbol{\theta}_0 \in \mathbb{R}^{3K+3}$ is the zero-pose vector. Using these pose corrections, we can write a *posed* face as

$$\mathbf{s}'(\boldsymbol{\beta}, \boldsymbol{\psi}, \boldsymbol{\theta}) = \mathbf{s}(\boldsymbol{\beta}, \boldsymbol{\psi}) + \Delta\mathbf{p}(\boldsymbol{\theta})\,, \tag{2.29}$$

on which a standard vertex-based linear blend skinning is applied. For appearance, linear texture models that build on the FLAME topology have been subsequently developed [176]. These models express textures as UV-space *maps* and are of the form

$$\mathbf{t} = \bar{\mathbf{t}} + \mathbf{B}_{\text{tex}}\boldsymbol{\phi}\,, \tag{2.30}$$

where $\bar{\mathbf{t}} \in \mathbb{R}^{|\mathbf{t}|}$ is the mean texture, $\mathbf{B}_{\text{tex}} \in \mathbb{R}^{|\mathbf{t}|\times|\boldsymbol{\phi}|}$ is the texture basis, and $\boldsymbol{\phi} \in \mathbb{R}^{|\boldsymbol{\phi}|}$ are the texture coefficients. For example, if the texture map is of resolution $r \times r$ and contains c channels, we have $|\mathbf{t}| = r \times r \times c$ with $|\phi| \ll |\mathbf{t}|$.

Variants. Thus far, we have looked into traditional methods for constructing linear 3DMMs, which typically rely on high-quality 3D face scans and employ principal component analysis (PCA) to derive low-dimensional bases that capture variations in identity, expression, and appearance. While these linear models have proven effective for encoding coarse geometric structure and moderately detailed texture, they exhibit limited capacity to represent fine-grained facial details—such as wrinkles, moles, freckles, or other subject-specific characteristics. Moreover, the diversity of face variations that a 3DMM can model is inherently constrained by the scope of the underlying 3D scan dataset, which is often expensive and logistically challenging to collect at a large scale with demographic balance. To mitigate the reliance on curated 3D scan datasets, a new class of methods has emerged that learns 3DMMs directly from in-the-wild images or videos. Among these, Tran and Liu [177, 178] replace linear PCA bases with nonlinear, convolutional decoders to enhance the capacity of the learned shape and texture spaces. Other approaches, such as TRUST [179], still utilize 3D scans but focus on increasing albedo diversity by incorporating a wider range of skin tones into the training data. More broadly, the fundamental premise of a 3DMM—learning a compact, parametric space that captures the natural distribution of human faces—has been extended beyond mesh-based representations to include alternative representations such as implicit surfaces [180], neural radiance fields [181], and 3D Gaussian Splatting [182], all of which offer new trade-offs in terms of fidelity, generalization, and rendering speed.

Having established the foundations of parametric face modeling, we now turn our attention to 3D face reconstruction algorithms, many of which incorporate these models as priors or constraints. Mirroring the taxonomy introduced in the previous section, our discussion is organized around the type of underlying representation employed by these algorithms. While this overview provides a broad perspective on the landscape of face reconstruction, our main focus is on static reconstructions from single images. We defer the discussion of dynamic reconstruction from single- or multi-view sequences and the animation of these reconstructions to subsequent chapters.

2.3.3 Surface-based Methods

Many face reconstruction approaches adopt geometry and appearance models (G, A) defined by a linear, mesh-based 3DMM, and estimate the model parameters $\mathbf{p}$ in an analysis-by-synthesis fashion [3]. Parameter estimation can be carried out either via direct optimization [183, 184] (*e.g.*, by minimizing the discrepancy between rendered and observed images) or by learning a function, typically implemented as a neural network, that maps input images to model parameters [185, 186]. However, reconstructions obtained through these methods are inherently restricted to the subspace spanned by the linear 3DMMs, often resulting in overly smooth or generic face geometry.

To enhance detail and capture finer structures, one line of work introduces a refinement stage following a coarse model-based reconstruction [187, 188]. Alternatively, model-free approaches abandon the parametric formulation entirely to allow greater flexibility [189, 190], while other methods learn more expressive, nonlinear parametric models that address the reconstruction problem simultaneously [177, 178, 191]. Another strategy for capturing fine-scale details involves encoding high-frequency geometry into texture maps, effectively *baking* geometric variations into appearance [192, 193]. Although this yields visually compelling results, these representations typically lack the physical correctness required for downstream tasks like relighting or animation. Animating reconstructed faces, *i.e.*, modifying control parameters $\mathbf{p}$ to synthesize novel expressions, also demands a careful algorithmic design. In particular, geometry deformations must account for person-specific traits such as expression lines, dimples, or asymmetric features to achieve photorealistic and identity-preserving animations.

It is worth noting that many mesh-based reconstruction algorithms rely on gradient-based optimization to estimate model parameters. As a result, these approaches require the rendering operation $\mathcal{R}$ to be differentiable with respect to geometry and appearance realizations as well as camera parameters. Furthermore, the underlying geometry and appearance *models* must also be differentiable with respect to their control parameters. Achieving this in discrete representations such as meshes is non-trivial, as it requires a careful re-implementation of each step throughout the rendering process to preserve differentiability. Nowadays, several differentiable mesh renderers are publicly available [194–196] and tightly integrated with modern deep learning frameworks. These tools have significantly lowered the barrier to implementing gradient-based mesh optimization and have inspired further research into converting non-differentiable rasterization pipelines to differentiable ones with minimal engineering [197].

Most surface-based face reconstruction methods represent geometry using a single mesh, and therefore, they are inherently limited by a finite resolution. To overcome this limitation, several approaches adopt continuous representations such as SDFs to enable resolution-independent reconstruction and recover fine-grained details that discrete meshes may miss [180, 198–200]. Further, native differentiability of some of these representations can facilitate integration into gradient-based learning pipelines without the need for specialized rendering implementations. Nevertheless, continuous surface-based methods, like their mesh-based counterparts, still model the face as a single surface. As a result, they also struggle to represent volumetric and complex structures such as hair, beards, translucent face regions, or self-occlusions of the face, such as the oral cavity. This limits the applicability of these models in high-fidelity scenarios that demand photorealistic 3D digital humans. Naturally, these constraints motivate a shift toward volumetric approaches, which we briefly explore next.

2.3.4 Volumetric Methods

For static faces, early volumetric reconstruction methods in the deep learning era employed convolutional neural networks to regress 3D voxel grids from images directly [201, 202]. The emergence of radiance field-based representations [49, 50] significantly enhanced the capabilities for high-quality volumetric modeling, enabling richer geometry and appearance reconstructions with view-consistent details [181, 203, 204].

In the absence of explicit 3D priors or multiview data, one can resort to 3D generative models learned from in-the-wild face images. With single-step models, such as those learned with an adversarial training framework [205], 3D reconstruction may involve inverting an image into the model's latent space [10, 11, 14, 206]. With multi-step models, such as diffusion models [89], it was demonstrated that 2D image or video models can capture useful 3D priors, eliminating the need for explicit 3D generative modeling [98, 207, 208]. A more recent line of work explores transformer-based large reconstruction models [209, 210] that are trained on massive datasets to learn powerful priors about the 3D reality. Via a single forward pass, these models can quickly predict 3D realizations of 2D observations. Although developed for general scene reconstruction, the underlying architecture and training paradigm of these models are well-suited for adaptation to face-specific reconstruction tasks using curated face datasets.

When single-view or multiview *video* datasets are available, the 3D reconstruction problem can be seen as a performance capture problem, which involves estimating temporally coherent, dynamic 3D faces. As research in this area has showcased compelling, photorealistic dynamic face synthesis with volumetric effects, the attention has shifted to building *expression-controllable* 3D face models—an inherently more complex challenge. This brings us to the realm of 3D photoreal *avatars*, which is the concluding focus of this thesis. To properly contextualize this progression, we postpone our discussion of the broader concept of 3D digital humans until Chapter 4, where we turn our attention to the *application-driven* aspects of the dynamic 3D face reconstruction *task*.

3

3D-Aware Face Image Manipulation

3.1 Introduction

Face image manipulation refers to the task of altering one or more attributes of a given face image. It has many prominent applications in the entertainment domain, where users can modify their portrait images according to personal preferences, such as smoothing skin textures, enhancing hair appearance, adjusting facial illumination to match a new background, or even adding virtual accessories. While these are undoubtedly popular use cases, which have transformed content creation, virtual try-ons, and portrait enhancement, face manipulation pipelines extend far beyond these scenarios [211]. For example, these pipelines can be employed to create synthetic datasets that augment existing ones, thereby enabling the learning of more robust face representations for downstream tasks [212]. Moreover, given that face images often carry sensitive biometric information, data collected without consent can be processed using face de-identification or anonymization techniques to safeguard privacy [213]. This diverse set of applications makes face editing algorithms a compelling research topic with several key technical challenges.

3.1.1 Main Challenges

Before devising a face manipulation algorithm, it is essential to first clarify what we mean by *manipulations*: which face attributes are to be considered, and how can a user describe a desired edit? Perhaps the most intuitive approach involves a natural language description, where the user provides a *prompt* directly to the algorithm. While such high-level commands may be convenient for end users, certain applications may demand a more physically-grounded and parameterized description of manipulations, enabling finer control and reproducibility. This may involve, for example, describing 3D face shape manipulations using an explicit 3D mesh or representing the illumination changes by parameterizing lighting conditions with a set of basis functions that describe view-dependent intensity and color variations. Historically, many methods have approached face manipulation using single-step pretrained generative models [205], often

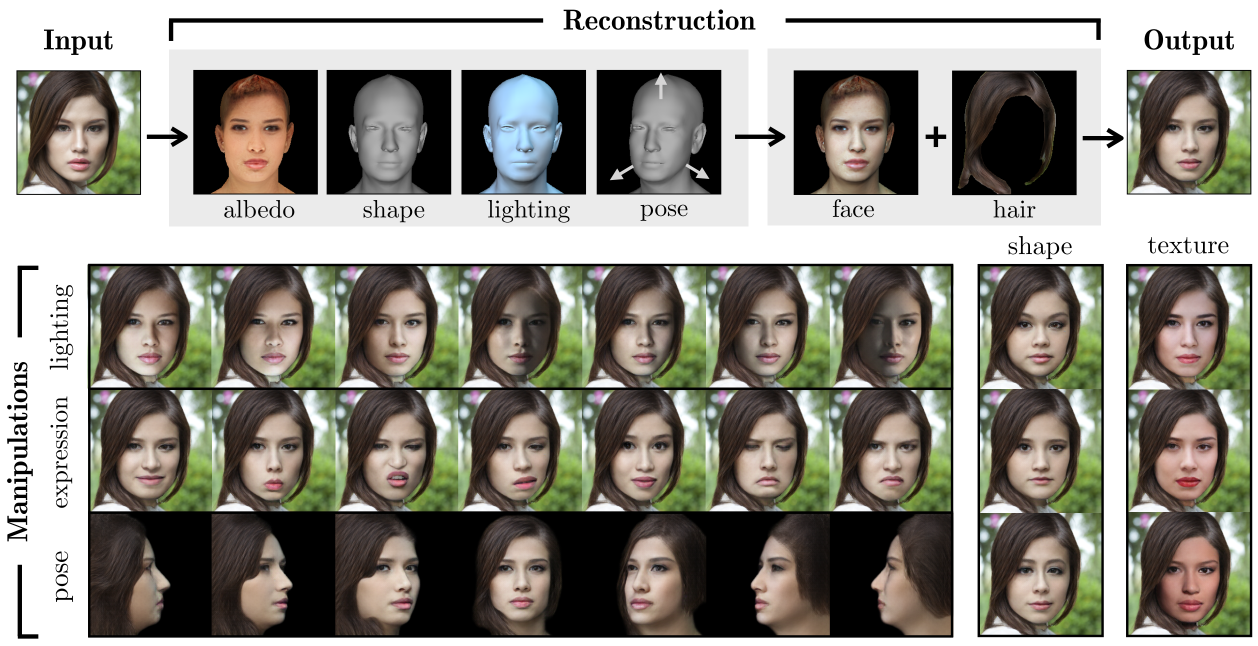

Figure 3.1: Overview of our fully disentangled, 3D controllable face image manipulation pipeline.

incorporating 3D Morphable Models (3DMMs) [3] as control priors. These approaches typically operate in the latent spaces of generative architectures, allowing for attribute modifications by traversing specific directions in these compact representations. Yet, these lower-dimensional descriptions of highly complex face images exhibit significant entanglement among distinct face attributes, such that altering a single attribute frequently induces unintended changes in others. Overcoming this, *i.e.*, achieving *disentanglement*, remains a fundamental and challenging objective.

Another consideration in face image manipulation is the degree of *3D-awareness* of the manipulation algorithm. For example, when changing lighting conditions, it may be desirable to generate cast shadows that accurately conform to the 3D geometry of the face. Similarly, when modifying expressions, such as enhancing a subtle smile into a broad grin, it may be important to reproduce secondary effects like skin wrinkling, nasolabial folds, and the shifting of details across the face. However, generative face models developed prior to the work presented in this chapter have predominantly relied on 2D-only architectures, largely neglecting explicit 3D reasoning during synthesis. As a result, while these models can implicitly encode 3D information in their latent spaces, their lack of explicit 3D representations limits their capacity to handle view-dependent effects and geometry-aware edits with high fidelity. This limitation highlights the need for models that not only synthesize visually appealing outputs but also possess a deeper understanding of the underlying 3D structure and photometric properties of human faces.

Computational efficiency of face manipulation algorithms is also an important factor, particularly for applications that require edits across large datasets. For example, tasks such as face anonymization across massive photo archives or videos may demand algorithms capable of processing thousands of faces in a short time frame. When manipulating a real face image, or in general, an arbitrary face image *not* generated by the pretrained face model at hand, early methods often resort to optimization-based techniques to embed input images into the latent space of these models via iterative procedures [19]. While these approaches have demonstrated success in photorealistic editing of real images, they are both computationally expensive and may fail to preserve fine-level details of the face, sometimes leading to perceptible artifacts. Designing algorithms that can take an input image and produce a photorealistic edited version of it in under a second on modern hardware remains a formidable yet desirable goal for practical deployment.

Nowadays, publicly available large-scale face image datasets typically lack sufficient variation in head poses, facial expressions, and lighting conditions, resulting in generative models biased toward a narrow distribution of *average* faces [84]. Consequently, when these models are tasked with performing extreme manipulations, such as rotating a face far from the frontal view, synthesizing uncommon expressions, or relighting faces under complex environments, they often fail to generalize and produce unrealistic outputs [19]. This calls for new architectures and learning paradigms capable of handling out-of-distribution scenarios with high visual fidelity, which can be crucial for certain face editing applications.

3.1.2 Main Objective and Contributions

In this chapter, we address the problem of photorealistic face image manipulation using a dataset of single-view, in-the-wild face images. Our goal is to develop an algorithm capable of tackling four key challenges: 1) disentanglement of face attributes, 2) 3D-awareness of manipulations, 3) computational efficiency, and 4) robustness to extreme manipulations beyond the training data distribution. Our proposed solutions to these challenges are outlined below. Please see Figure 3.1 for an overview of our pipeline.

Disentanglement. By explicitly estimating and processing each face attribute independently within the architecture, our method achieves full disentanglement by design. Such an architecture ensures that modifications to one attribute do not inadvertently affect others, enabling precise and semantically consistent edits.

3D-awareness. We introduce a fully 3D-aware architecture that maintains computational efficiency at inference time. This is accomplished by estimating 3D geometry and appearance parameters using *2D* convolutional operations, which are then mapped onto 3D assets via a predefined template mesh topology. This approach leverages the efficiency of 2D convolutions while preserving explicit 3D structural information.

Computational efficiency. To replace computationally expensive optimization procedures, commonly used to embed input images into latent spaces, we design a learnable function that performs this operation in a single forward pass. Specifically, we employ separate encoders for each face attribute, allowing direct estimation of controllable 3D representations from an input image.

Support for extreme manipulations. A key insight in our approach lies in representing face attributes in their physical spaces, without resorting to compressed spaces or lower-dimensional abstractions. By preserving full physical representations of geometry, appearance, and lighting, our pipeline showcases generalization to manipulations outside the training data distribution.

3.1.3 Preliminaries

In this part of the thesis, we consider the following variation of the observation model in (2.25):

$$\mathbf{y} = \Psi(\mathcal{R}(G(\mathbf{p}), A(\mathbf{p}); \mathbf{c})) + \mathbf{n}\,, \tag{3.1}$$

where $\Psi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ denotes an image enhancement function applied to the rendered 2D output. This post-processing step can be beneficial when the underlying geometric and appearance models fall short of capturing the full complexity of the 3D reality, as is often the case with single-surface representations, which we adopt in this work. Given a *single* face image, our objective is to render it under different control inputs such as 3D shape, albedo, or lighting. To achieve this, we assume that we are given a dataset of N single-view face images of different subjects $\{\mathbf{y}_i\}_{i=1}^N$ without ground truth camera parameters. Since we do not know the camera parameters nor the image enhancement operation Ψ, we have a *blind* inverse problem with a complex but known, nonlinear rendering operation $\mathcal{R}$. We assume these operations to be differentiable as we aim to recover a 3D face using gradient-based optimization algorithms. In what follows, we use notation consistent with the published work [214].

3.2 Background and Related Work

To lay the groundwork for our pipeline, we first review the principles of generative adversarial networks [205], highlighting the architectural components critical to our approach. Then, we provide a literature review of face image manipulation methods that precede our methodology, where we discuss our method's key differences with these modern frameworks.

3.2.1 Generative Adversarial Networks

To build the principles of generative adversarial networks (GANs) [205], we begin by deriving the adversarial training framework, including its theoretical formulation and the optimality conditions of the minimax objective central to GANs. We then narrow our focus to a specific class of generator architectures, namely style-based models [84], which have proven highly effective in synthesizing photorealistic images and form the backbone of our proposed approach.

3.2.1.1 Derivation of Adversarial Training

Given a set of samples drawn from an unknown data distribution, the goal of *generative modeling* is to approximate this underlying distribution with a probabilistic model that faithfully captures its statistics. Generative models form a cornerstone of modern machine learning, enabling a range of capabilities including the synthesis of novel samples and the construction of data-driven priors that can be used in solving inverse problems, representation learning, coding, and compression [215, 216]. For visual signals—such as images, videos, or 3D assets—the generative modeling problem becomes especially challenging due to the high dimensionality and intricate structure of the underlying distributions, which often exhibit complex dependencies that are difficult to describe analytically. A common approach to generative modeling involves learning a deterministic function, parameterized by weights $\mathbf{w}_g$, that maps a sample from a simple prior distribution $p_{\mathbf{z}}$ (*e.g.*, a standard Gaussian) to a sample in the target data space:

$$G(\cdot;\mathbf{w}_g) : \mathbb{R}^{|\mathbf{z}|} \longrightarrow \mathbb{R}^{|\mathbf{x}|}, \tag{3.2}$$

which takes in a latent vector $\mathbf{z} \in \mathbb{R}^{|\mathbf{z}|}$ distributed according $p_{\mathbf{z}}$ to and *generates* a sample $\mathbf{x} \in \mathbb{R}^{|\mathbf{x}|}$. Let p_m denote the *derived* distribution that characterizes the output $\mathbf{x}$ of the model G. Given a dataset of samples $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$, each distributed according to an unknown p_d, generative modeling aims to match the model distribution p_m to the data distribution p_d by optimizing the generator weights $\mathbf{w}_g$. To achieve this, the *adversarial training framework* [205] leverages another trainable function parameterized by weights $\mathbf{w}_d$:

$$D(\cdot;\mathbf{w}_d) : \mathbb{R}^{|\mathbf{x}|} \longrightarrow [0,1], \tag{3.3}$$

which takes in a sample $\mathbf{x} \in \mathbb{R}^{|\mathbf{x}|}$ and outputs a scalar $s \in [0,1]$ that represents the probability that the given sample originates from the *real* distribution p_d. This function, referred to as the *discriminator*, is trained to differentiate between *real* samples drawn from the dataset and *fake* samples synthesized by a generator. In parallel, the generator G is optimized to produce samples that are indistinguishable from real data, thereby fooling the discriminator D into classifying generated instances as real. This adversarial training paradigm can be formalized as a two-player minimax game, defined over a value function $V(G,D)$:

$$\min_G \max_D V(G,D) = \min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_d}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1-D(G(\mathbf{z})))], \tag{3.4}$$

which takes the form of a binary cross-entropy objective. For any given generator G, we can expand $V(G,D)$ as follows:

$$\begin{aligned} V(G,D) &= \int_{\mathbf{x}} p_d(\mathbf{x})\ \log(D(\mathbf{x}))d\mathbf{x} + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z})\ \log(1-D(G(\mathbf{z}))d\mathbf{z} && (3.5)\\ &= \int_{\mathbf{x}} [p_d(\mathbf{x})\ \log(D(\mathbf{x})) + p_m(\mathbf{x})\log(1-D(\mathbf{x}))]d\mathbf{x}, && (3.6)\end{aligned}$$

where we apply the change of variable $\mathbf{x} = G(\mathbf{z})$ to the expectation over $\mathbf{z}$. If we define $f(D) := p_d \log D + p_m \log(1-D)$, one can verify that $f(D)$ is maximized at $D^* = p_d/(p_d+p_m)$, and hence $D^*(\mathbf{x}) = p_d(\mathbf{x})/[p_d(\mathbf{x})+p_m(\mathbf{x})]$ for every $\mathbf{x}$. Plugging this into the objective, we have

$$\begin{aligned} V(G,D^*) &= \int_{\mathbf{x}} \left[p_d(\mathbf{x})\ \log\frac{p_d(\mathbf{x})}{p_d(\mathbf{x})+p_m(\mathbf{x})} + p_m(\mathbf{x})\ \log\frac{p_m(\mathbf{x})}{p_d(\mathbf{x})+p_m(\mathbf{x})}\right] d\mathbf{x} && (3.7)\\ &= \int_{\mathbf{x}} \left[(p_d(\mathbf{x})+p_m(\mathbf{x}))\log\frac{1}{2} + p_d(\mathbf{x})\log\frac{p_d(\mathbf{x})}{\frac{p_d(\mathbf{x})+p_m(\mathbf{x})}{2}} + p_m(\mathbf{x})\log\frac{p_m(\mathbf{x})}{\frac{p_d(\mathbf{x})+p_m(\mathbf{x})}{2}}\right] d\mathbf{x} && (3.8)\\ &= -\log 4 + D_{\mathrm{KL}}\left(p_d \,\middle\|\, \frac{p_d+p_m}{2}\right) + D_{\mathrm{KL}}\left(p_m \,\middle\|\, \frac{p_d+p_m}{2}\right) && (3.9)\\ &= -\log 4 + 2D_{\mathrm{JS}}\left(p_d \,\|\, p_m\right), && (3.10)\end{aligned}$$

where D_{KL} is Kullback-Leibler divergence and D_{JS} is Jensen-Shannon divergence defined as

$$D_{\mathrm{JS}}(p\|q) := \frac{1}{2}\left[D_{\mathrm{KL}}\left(p \,\middle\|\, \frac{p+q}{2}\right) + D_{\mathrm{KL}}\left(q \,\middle\|\, \frac{p+q}{2}\right)\right]. \tag{3.11}$$

Since these divergences are non-zero quantities, $V(G, D^*)$ is *minimized* when $p_d = p_m$ and achieves a constant value of $-\log 4$. When this is the case, the optimal discriminator $D^*(\mathbf{x}) = 1/2$ for all $\mathbf{x}$, meaning that the discriminator cannot do better than random guessing once the generator's distribution matches the data distribution. It can be shown that this solution is a Nash equilibrium of the minimax game in Equation 3.4 [217].

To achieve the Nash equilibrium, the original work [205] proposes a stochastic gradient-based algorithm that sequentially maximizes and minimizes the value function to update the discriminator and the generator, respectively. In practice, the objective is often high-dimensional and non-convex, and the proposed optimization algorithm poses stability challenges such as non-convergence or mode collapse [218]. While numerous works have proposed alternatives to the minimax objectives, novel training paradigms, or various heuristics and tricks [219–223], adversarial training remains a relatively unstable algorithm for generative modeling compared to more recent generative paradigms [89]. Nevertheless, the *generative adversarial network (GAN)* objective can be conveniently augmented in many deep learning-based approaches by introducing a single loss function, sometimes referred to as a *realism loss*, which promotes the realistic synthesis of different classes of signals. In fact, we pursue a similar objective in our photorealistic image manipulation pipeline that is developed later in this chapter.

3.2.1.2 Style-based Models for Image Synthesis

Up to this point, we have considered a generic data distribution p_d and arbitrary functions for the generator G and the discriminator D. We now narrow our focus to specific classes of images, whose structure guides the architectures of G and D. Given a vector $\mathbf{z}$ in some latent space $\mathcal{Z}$, one common approach is to employ a fully convolutional generator that maps the latent space to the image domain $\mathcal{X}$ (*i.e.*, $G : \mathcal{Z} \to \mathcal{X}$) along with a convolutional discriminator that maps images to scalar outputs, $D : \mathcal{X} \to [0, 1]$. Conventional generator designs typically inject the latent code only at the input layer, but this limits the model's ability to disentangle and control high-level semantic attributes in the generated images [84].

The seminal StyleGAN architecture [84] introduces a novel generator design, drawing inspiration from advances in the style transfer literature. To promote disentanglement, *i.e.*, to learn a latent space composed of linear subspaces that control a single semantic attribute, StyleGAN employs a learned nonlinear mapping $f : \mathcal{Z} \to \mathcal{W}$ that transforms latent vectors from the input space $\mathcal{Z}$ into an intermediate latent space $\mathcal{W}$ via an MLP. While samples $\mathbf{z} \in \mathcal{Z}$ are drawn from a simple, fixed prior (typically a standard Gaussian), their transformed counterparts $\mathbf{w} = f(\mathbf{z})$ exhibit a more complex distribution, empirically shown to yield more disentangled representations. These intermediate codes $\mathbf{w}$ are then used to condition a convolutional generator that synthesizes high-quality images that approximate the target data distribution.

The proposed generator in StyleGAN [84] transforms a learned constant tensor to an image through a number of convolutional and upsampling layers. To control the generator output, the intermediate vector $\mathbf{w}$ conditions each convolutional layer via an adaptive instance normalization (AdaIN) operation [224]. Specifically, for each layer ℓ, the vector $\mathbf{w}$ is first mapped to a pair of style vectors $\mathbf{y}_\ell^s, \mathbf{y}_\ell^b \in \mathbb{R}^{N_\ell}$ via learned affine transformations, where N_ℓ is the number of feature maps at layer ℓ. Then, the i-th feature map at this layer $\mathbf{x}_{\ell,i}$ is transformed according to

$$\bar{\mathbf{x}}_{\ell,i} = \mathbf{y}_{\ell,i}^s \frac{\mathbf{x}_{\ell,i} - \mu(\mathbf{x}_{\ell,i})}{\sigma(\mathbf{x}_{\ell,i})} + \mathbf{y}_{\ell,i}^b \tag{3.12}$$

where $\mathbf{y}^s_{\ell,i}, \mathbf{y}^b_{\ell,i} \in \mathbb{R}$ are the i-th component of the style vectors, and $\mu(\cdot), \sigma(\cdot)$ compute the mean and standard deviation across the elements of a vector, respectively. This style-based conditioning mechanism not only significantly enhances synthesis quality but also enables fine-grained control over the generated outputs across multiple scales, from coarse structural attributes to finer details.

Since its introduction, the original StyleGAN architecture has undergone several refinements to mitigate characteristic artifacts in the synthesized images, leading to subsequent iterations [225, 226]. StyleGAN2 [225] introduces modest architectural modifications, notably replacing instance normalization with weight demodulation, which effectively suppresses blob-like artifacts while preserving disentangled semantic control. Building on this, StyleGAN3 [226] presents a more substantial redesign of the generator to address the so-called texture sticking phenomenon, wherein high-frequency details erroneously lock to specific image coordinates, particularly during latent space interpolations or spatial transformations. In this chapter, we leverage the StyleGAN2 [225] architecture to synthesize textures for reconstructed 3D face meshes.

3.2.2 Face Image Manipulation

Photorealistic face image manipulation is inherently tied to the ability to synthesize realistic face images. In late 2010s, GANs [205] have established a new benchmark in high-quality image generation, with style-based variants [84, 225, 226] producing face images that are often indistinguishable from real photographs. While conventional GANs operate purely in the 2D domain, several methods have explored extending generative modeling to 3D by leveraging voxel-based [125, 227–233] and mesh-based [234–236] representations to better capture the underlying 3D structure. More recently, neural implicit representations have enabled continuous and differentiable 3D scene synthesis, including that of human faces [237, 238]. Parallel efforts have focused on extracting 3D-aware features from pretrained 2D GANs to support image manipulation in 3D [239, 240] or to recover explicit 3D geometry from single images [241, 242]. However, many of these methods lack strong geometric priors, which constrain their ability to perform precise 3D-aware manipulations. In contrast, our approach is grounded in a 3D architecture that explicitly incorporates the StyleGAN2 generator [225]. Notably, our method is trained without access to real, high-quality 3D data, yet it learns strong 3D face priors to achieve photorealistic synthesis and manipulation.

A central application of 3DMMs is the reconstruction of 3D faces from 2D images, with the goal of recovering either face shape alone [243, 244] or both shape and albedo [245, 246]. Approaches that jointly estimate shape and albedo have increasingly leveraged GANs to produce higher quality and more photorealistic textures [192, 193, 247]. Among these works, GANFIT [192] and AvatarMe [247] achieve reconstructions with rich high-frequency details, but their success depends on access to large-scale, high-quality 3D training datasets. Unlike these methods, our approach does not rely on high-quality 3D ground truth for supervision. Instead, we learn to synthesize photorealistic 3D face representations directly from 2D images. While several other methods also attempt 3D face recovery from 2D images [186, 246], their outputs lack the photorealism and miss important features of the face, such as hair or teeth.

A well-established line of research on 3D face modeling builds on 3DMMs [3, 9] to construct parametric models of face geometry and appearance from high-quality 3D scans. Conventional linear 3DMMs, such as the Basel Face Model [173, 248] and FLAME [174], are typically limited in expressiveness due to their reliance on low-rank PCA bases and the scarcity of diverse training data. These models often fail to capture fine-grained facial detail and realistic appearance variations. To overcome these limitations, several works have proposed nonlinear extensions of 3DMMs. Notably, [177, 178, 191] introduce deep learning-based models that significantly improve reconstruction quality compared to their linear counterparts. Neural architectures have also been explored for high-quality texture synthesis in various face modeling tasks [193, 249, 250], demonstrating the potential of learned representations to achieve better realism. In this chapter, we build our face model as a nonlinear 3DMM based on the FLAME topology. While FLAME's original linear bases lack the capacity to produce photorealistic outputs, we leverage them to generate synthetic training data and to impose regularization during albedo reconstruction. Importantly, although our model adopts the FLAME mesh template, it does not inherit FLAME's representational constraints: we learn an entirely new model with significantly enhanced expressiveness in both shape and appearance. Unlike prior work such as Tran and Liu [177], we employ separate encoders for different face attributes to promote disentanglement, and we integrate StyleGAN2 for albedo synthesis, enabling more photorealistic texture generation.

Recent efforts have explored the integration of 3D Morphable Models (3DMMs) into adversarial training frameworks to enable disentangled editing of portrait images [18–20, 251–257]. Among these, DiscoFaceGAN [18] employs contrastive learning to enforce attribute disentanglement, while StyleRig [19] couples a 3DMM with a pretrained 2D style-based generator and manipulates images in the latent space. However, both approaches rely on 2D generative models and are thus limited in handling challenging 3D variations such as extreme pose, illumination, and facial expression. To edit real face images, these methods typically require projection into the latent space of the generator via techniques such as Image2StyleGAN [258], which can degrade the quality of the results. To address this, Portrait Image Embedding (PIE) [20] introduces an optimization-based inversion scheme that better preserves photorealism during latent space projection. Nevertheless, as both StyleRig and PIE are built upon a pretrained 2D style-based generator, they are limited to the distribution of variations seen in the face image dataset [84] used to train this generator. Furthermore, since the disentanglement in these frameworks is introduced *post hoc*, they fall short of achieving fully independent control over physical attributes. Concurrent to the work presented in this chapter, GAR [255] introduces a photorealistic face reconstruction method that can be used to manipulate portrait images, and VariTex [254] introduces a variational texture model that enables face synthesis with attribute control. However, both methods primarily focus on manipulating expression and head poses. In contrast, we are more generally interested in modifying 3D shape, albedo, and lighting conditions.

3.3 Methodology

Our approach combines a statistical model of 3D faces with a style-based GAN, achieving a photorealistic and fully disentangled 3D model of faces. We achieve such disentanglement by individually processing each of the face's physical attributes in our architecture, through separate encoders and decoders, as illustrated in Figure 3.2. Such explicit control enables us to extrapolate beyond what is well-represented in the training set, allowing for face synthesis in extreme facial expressions and lighting conditions.

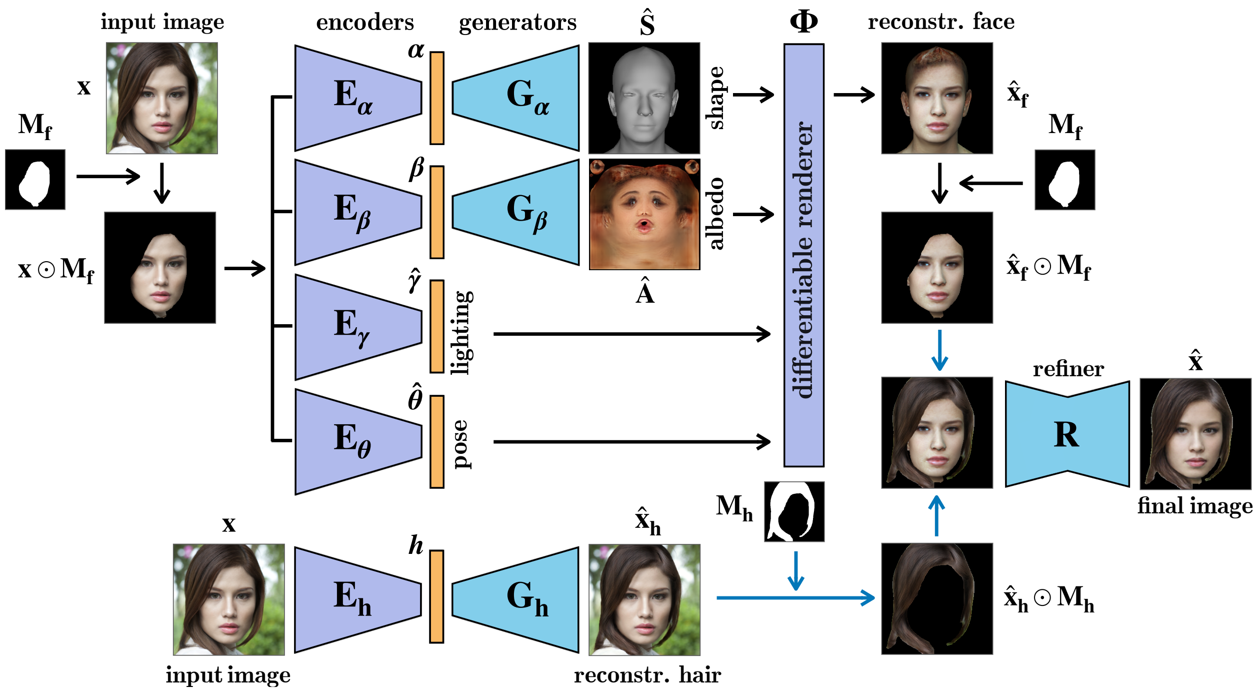

Figure 3.2: **Overview of our architecture.** Our model starts with a set of encoders $\{E_\alpha, E_\beta, E_\gamma, E_\theta, E_h\}$ for shape, albedo, lighting, pose, and hair, respectively. To reconstruct the shape and albedo in their physical spaces, we use a convolutional generator G_α for shape and a style-based generator G_β for albedo. The reconstructed face image $\hat{x}_f$ is produced using a differentiable renderer Φ. In addition to our face model, which is demarcated by black connecting arrows, a hair generator G_h reconstructs the hair in 2D. Reconstructed face and hair are finally fused and enhanced using a refiner network. All components are trained end-to-end, except for hair, where we deploy a pretrained, off-the-shelf hair model [259].

3.3.1 Problem Formulation

Our method primarily relies on reconstructing high-fidelity and photorealistic 3D faces from 2D images, following our formulation in Equation 3.1. First, we assume that a face image can be decomposed into five different attributes: four *physical attributes* (3D shape, albedo, lighting, and pose) and hair. Here, the physical attributes collectively define the unknown 3D *realization* of a 2D face image. To construct our face model, we define a set of encoders $\{E_\alpha, E_\beta, E_\gamma, E_\theta\}$ that individually map an input image to each of the physical attributes to promote disentanglement. To mask out the regions that cannot be represented by a mesh-based 3D face model [174], we use an off-the-shelf segmentation model [260] to segment the input image $\mathbf{x}$ and subsequently obtain $\mathbf{x}' := \mathbf{x} \odot M_f$, where M_f denotes the mask. The encoders E_α and E_β extract a latent shape code $\boldsymbol{\alpha}$ and albedo code $\boldsymbol{\beta}$, while E_γ and E_θ directly estimate the lighting parameters $\hat{\boldsymbol{\gamma}}$ and pose parameters $\hat{\boldsymbol{\theta}}$. To generate a face image, the shape and albedo codes are fed to a shape generator G_α and albedo generator G_β, respectively, to produce a 3D shape $\hat{S}$ and albedo map $\hat{A}$, which by design have a higher representational capacity than a linear 3DMM. Next, a differentiable renderer Φ renders the generated 3D model $\{\hat{S}, \hat{A}\}$ using the lighting and pose parameters $\{\hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\theta}}\}$ to produce the reconstructed face $\hat{\mathbf{x}}_f = \Phi(\hat{S}, \hat{A}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\theta}})$. A discriminator $\mathbf{D}$, not shown in Figure 3.2, is employed to enhance photorealism through adversarial training.

Since our face model is built on a 3DMM that does not model hair, we couple it with an explicit 2D hair model, which consists of an encoder $\mathbf{E_h}$ and a generator $\mathbf{G_h}$ to produce a portrait image with reconstructed hair $\hat{\mathbf{x}}_\mathbf{h}$. The outputs of the face model and the hair model are combined using a face mask $\mathbf{M_f}$ and a hair mask $\mathbf{M_h}$, then passed through a refiner network $\mathbf{R}$ that produces the final image $\hat{\mathbf{x}}$. Formally, given a set of N portrait images along with their face masks and hair masks $\{(\mathbf{x}^i, \mathbf{M}_\mathbf{f}^i, \mathbf{M}_\mathbf{h}^i)\}_{i=1}^N$, our objective is to solve the following optimization problem:

$$\underset{\{\mathbf{E}_\alpha,\mathbf{E}_\beta,\mathbf{E}_\gamma,\mathbf{E}_\theta,\mathbf{G}_\alpha,\mathbf{G}_\beta,\mathbf{R}\}}{\arg\min} \sum_{i=1}^{N} \|\mathbf{x}^i \odot (\mathbf{M}_\mathbf{f}^i + \mathbf{M}_\mathbf{h}^i) - \hat{\mathbf{x}}^i\|_1 \tag{3.13}$$

where $\hat{\mathbf{x}} = \mathbf{R}(\hat{\mathbf{x}}_\mathbf{f} \odot \mathbf{M_f} + \hat{\mathbf{x}}_\mathbf{h} \odot \mathbf{M_h})$, with $\hat{\mathbf{x}}_\mathbf{f} = \mathbf{\Phi}(\mathbf{G}_\alpha(\mathbf{E}_\alpha(\mathbf{x}')), \mathbf{G}_\beta(\mathbf{E}_\beta(\mathbf{x}')), \mathbf{E}_\gamma(\mathbf{x}'), \mathbf{E}_\theta(\mathbf{x}'))$ and $\hat{\mathbf{x}}_\mathbf{h} = \mathbf{G_h}(\mathbf{E_h}(\mathbf{x}))$. In later sections, we show that adopting this objective enables us to edit portrait images in a fast, fully disentangled manner while preserving their photorealism.

3.3.2 Face Model

Our face model consists of four physical attribute encoders, two generators, and a differentiable renderer [194]. In the shape pipeline, the shape code $\boldsymbol{\alpha}$ is input to a convolutional generator, $\mathbf{G}_\alpha$. The generated 3D shape, $\hat{\mathbf{S}}$, is composed of 3 channels in the UV-space that represent the 3D coordinates of vertices [177] by their displacement from the FLAME mean head model [174]. In parallel, the albedo code $\boldsymbol{\beta}$ goes through a StyleGAN2 [225] generator $\mathbf{G}_\beta$ that outputs an RGB albedo map $\hat{\mathbf{A}}$ in the UV-space. Since most of the variations in face images are due to the variations in the albedo, generating albedo with a style-based architecture is a crucial step to achieve realism in the final output. Furthermore, in order to allow for more expressive latent spaces of shape and albedo, we let our model learn them without being constrained to the subspace defined by the original 3DMM. Finally, we represent the estimated lighting $\hat{\boldsymbol{\gamma}}$ using a spherical harmonics parameterization with 3 bands [143, 261], and our 6-DOF pose vector $\hat{\boldsymbol{\theta}}$ includes 3 parameters for 3D rotation using the axis-angle representation and 3 parameters for 3D translation.

We divide our training process into two stages: 1) we pretrain our face model on synthetically generated faces; then 2) we generalize our model to real faces by training on real 2D images. The loss functions for each stage are introduced in the equations below:

Synthetic data Pretraining

$$L_{\text{image}}^{\text{syn}} = \|\mathbf{x} - \hat{\mathbf{x}}_\mathbf{f}\|_2^2 \tag{3.14}$$
$$L_{\text{albedo}}^{\text{syn}} = \|\mathbf{A} - \hat{\mathbf{A}}\|_2^2 \tag{3.15}$$
$$L_{\text{shape}}^{\text{syn}} = \|\mathbf{w_s}^T(\mathbf{S} - \hat{\mathbf{S}})\|_2^2 \tag{3.16}$$
$$L_{\text{pose}}^{\text{syn}} = \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2^2 \tag{3.17}$$
$$L_{\text{lighting}}^{\text{syn}} = \|\boldsymbol{\gamma} - \hat{\boldsymbol{\gamma}}\|_2^2 \tag{3.18}$$
$$L_{\text{reg}}^{\text{syn}} = \lambda_\alpha\|\boldsymbol{\alpha}\|_2^2 + \lambda_\beta\|\boldsymbol{\beta}\|_2^2 \tag{3.19}$$
$$L_{\text{gan}}^{\text{syn}} = -\log \mathbf{D}(\hat{\mathbf{x}}_\mathbf{f}) \tag{3.20}$$

Real data Training

$$L_{\text{image}}^{\text{real}} = \|\mathbf{x} \odot \mathbf{M_f} - \hat{\mathbf{x}}_\mathbf{f} \odot \mathbf{M_f}\|_2^2 \tag{3.21}$$
$$L_{\text{id}}^{\text{real}} = 1 - \cos(f_{\text{id}}(\mathbf{x}),\ f_{\text{id}}(\hat{\mathbf{x}}')) \tag{3.22}$$
$$L_{\text{lmk}}^{\text{real}} = \|\mathbf{w_l}^T[f_{\text{lmk}}^{(1)}(\mathbf{x}) - f_{\text{lmk}}^{(2)}(\hat{\mathbf{S}})]\|_2^2 \tag{3.23}$$
$$L_{\text{albedo}}^{\text{real}} = \|(\mathbf{B}^\mathbf{T}\mathbf{B})^{-1}\mathbf{B}^\mathbf{T}(\hat{\mathbf{A}} - \bar{\mathbf{A}})\|_2^2 \tag{3.24}$$
$$L_{\text{lighting}}^{\text{real}} = (\hat{\boldsymbol{\gamma}} - \bar{\boldsymbol{\gamma}})^T\mathbf{\Sigma}^{-1}(\hat{\boldsymbol{\gamma}} - \bar{\boldsymbol{\gamma}}) \tag{3.25}$$
$$L_{\text{reg}}^{\text{real}} = \lambda_\alpha\|\boldsymbol{\alpha}\|_2^2 + \lambda_\beta\|\boldsymbol{\beta}\|_2^2 \tag{3.26}$$
$$L_{\text{gan}}^{\text{real}} = -\log \mathbf{D}(\hat{\mathbf{x}}_\mathbf{f} \odot \mathbf{M_f}) \tag{3.27}$$

Pretraining on Synthetic Data. The first stage is a pretraining step to allow our network to capture important characteristics of faces using strong supervision coming from a linear 3DMM. In this stage, we use the FLAME model to sample 80 000 faces under an illumination and pose prior [18]. We translate each face in 3D so that the rendered faces have the same 2D alignment as the ones in the real dataset [84]. Although these synthetic faces lack realism, they have ground truth values for the disentangled physical attributes albedo $\mathbf{A}$, shape $\mathbf{S}$, pose $\boldsymbol{\theta}$, and lighting $\boldsymbol{\gamma}$, which we use to guide pretraining. Our loss function for pretraining consists of three parts: reconstruction losses for the reconstructed face image (3.14) and for the four physical attributes (3.15)–(3.18); regularization for shape and albedo codes (3.19); and a non-saturating logistic GAN loss [217] to improve photorealism (3.20). In the shape reconstruction loss (3.16), we introduce a weighting term $\mathbf{w_s}$ to upweight vertices in regions surrounding salient facial features (e.g., eyes, eyebrows, mouth). We carry out pretraining in three independent phases: albedo-only, lighting-only, and shape & pose jointly. We minimize the following loss functions for the three phases:

$$\text{shape \& pose:} \qquad 0.1L_{\text{image}}^{\text{syn}} + 1000L_{\text{shape}}^{\text{syn}} + 100L_{\text{pose}}^{\text{syn}} + 1.0\|\boldsymbol{\alpha}\|_2^2 \tag{3.28}$$

$$\text{albedo-only:} \qquad L_{\text{gan}}^{\text{syn}} + 10L_{\text{image}}^{\text{syn}} + 100L_{\text{albedo}}^{\text{syn}} + 1.0\|\boldsymbol{\beta}\|_2^2 \tag{3.29}$$

$$\text{lighting-only:} \qquad 10L_{\text{image}}^{\text{syn}} + 100L_{\text{light}}^{\text{syn}} \tag{3.30}$$

For each of these phases, we set the batch size to 16 and use the Adam optimizer [262]. $\mathbf{E}_{\boldsymbol{\alpha}}$, $\mathbf{E}_{\boldsymbol{\beta}}$, $\mathbf{E}_{\boldsymbol{\gamma}}$, $\mathbf{E}_{\boldsymbol{\theta}}$, and $\mathbf{G}_{\boldsymbol{\alpha}}$ are all trained with a learning rate of 0.0001. During the albedo-only phase, we alternate optimization steps between training the albedo generator $\mathbf{G}_{\boldsymbol{\beta}}$ and an image discriminator $\mathbf{D}$ (both with learning rate 0.002), where the discriminator is trained to minimize the loss:

$$L_{\text{disc}}^{\text{syn}} = -\frac{1}{2}\log \mathbf{D}(\mathbf{x}) - \frac{1}{2}\log(1 - \mathbf{D}(\hat{\mathbf{x}}_{\mathbf{f}})). \tag{3.31}$$

Training on Real Data. After pretraining, we train our model using the FFHQ face dataset [84] at 256×256 resolution. We obtain the face mask $\mathbf{M_f}$ for each image automatically using a semantic segmentation network [260, 263], then feed the masked 2D face images to the network. We train our face model in an end-to-end fashion, where we combine the loss functions in (3.21)–(3.26) with a non-saturating logistic GAN loss (3.27). Since we do not know the ground truth physical attributes for the real face images, we cannot apply any of the physical attribute reconstruction losses (3.15)–(3.18). The only reconstruction loss we apply is a pixelwise reconstruction loss for the masked faces (3.21). Defining the full reconstructed image as $\hat{\mathbf{x}}' := \mathbf{x} \odot (1 - \mathbf{M_f}) + \hat{\mathbf{x}} \odot \mathbf{M_f}$, we impose an identity loss (3.22), where $f_{\text{id}}(\cdot)$ denotes the feature vector extracted by the ArcFace face recognition network [264], and $\cos(\cdot, \cdot)$ denotes cosine similarity. Our landmark loss (3.23) measures the distance between the image-plane projections of the 3D facial landmark locations in the input image (estimated using [265]) and the corresponding locations in the reconstructed 3D shape model. The shape model vertices corresponding to specific facial landmarks are defined by the FLAME topology, and the weighting term $\mathbf{w_l}$ places more weight on important landmarks such as the lip outlines to keep our learned model faithful to the FLAME topology.

Since the decomposition of an input image into physical face properties is an ill-posed problem, there are ambiguities such as the relative contributions of color lighting intensities and surface albedo to the RGB appearance of a skin pixel. To help resolve this ambiguity, we introduce an albedo regularization loss (3.24) to minimize the projection of our reconstructed albedo into the

FLAME model's albedo PCA space. Here, $\bar{\mathbf{A}}$ and $\mathbf{B}$ respectively represent the mean and basis vectors of the albedo model. To address the same ambiguity, we also include a lighting regularization loss (3.25), which maximizes the log-likelihood of the reconstructed lighting parameters $\hat{\boldsymbol{\gamma}}$ under a multivariate Gaussian distribution over lighting conditions. To obtain that distribution, we sample 50 000 lighting vectors using the prior provided by [18] and calculate their sample mean $\bar{\boldsymbol{\gamma}}$ and sample covariance $\boldsymbol{\Sigma}$. As in pretraining, (3.26) regularizes the shape and albedo codes.

During training, we split the FFHQ dataset [84] into train and test sets with 90% – 10% split, using the first 63 000 face images for training and the last 7 000 images for testing. We optimize over all networks in our face model ($\mathbf{E}_{\boldsymbol{\alpha}}, \mathbf{E}_{\boldsymbol{\beta}}, \mathbf{E}_{\boldsymbol{\gamma}}, \mathbf{E}_{\boldsymbol{\theta}}, \mathbf{G}_{\boldsymbol{\alpha}}, \mathbf{G}_{\boldsymbol{\beta}}$) jointly in an end-to-end fashion, and we alternate optimization steps between the face model and an image discriminator $\mathbf{D}$. For the first 50 000 iterations, we minimize the following loss function for all blocks in the face model, using a batch size of 16 and the Adam optimizer with learning rate 10^{-5}:

$$L_{\text{gan}}^{\text{real}} + 10^3\, L_{\text{image}}^{\text{real}} + 10\, L_{\text{identity}}^{\text{real}} + 10^2\, L_{\text{landmark}}^{\text{real}} + L_{\text{albedo}}^{\text{real}} + 10^{-5}\, L_{\text{lighting}}^{\text{real}} + \|\boldsymbol{\alpha}\|_2^2 + \|\boldsymbol{\beta}\|_2^2 . \quad (3.32)$$

Using a batch size of 16 and a learning rate of 10^{-5}, we train the discriminator by minimizing

$$L_{\text{disc}}^{\text{real}} = -\frac{1}{2}\log \mathbf{D}(\mathbf{x} \odot \mathbf{M_f}) - \frac{1}{2}\log(1 - \mathbf{D}(\hat{\mathbf{x}}_{\mathbf{f}} \odot \mathbf{M_f})). \quad (3.33)$$

After training our face model for 50 000 iterations, we fine tune $\mathbf{E}_{\boldsymbol{\beta}}, \mathbf{E}_{\boldsymbol{\gamma}}, \mathbf{G}_{\boldsymbol{\beta}}$ for another 50 000 iterations by freezing the weights of $\mathbf{E}_{\boldsymbol{\alpha}}, \mathbf{E}_{\boldsymbol{\theta}}, \mathbf{G}_{\boldsymbol{\alpha}}$ and discarding the landmark loss to further improve the reconstruction quality.

3.3.3 Hair Model

Since hair has a more complex structure than faces, representing and manipulating hair in 3D is a very challenging problem. This motivates us to manipulate hair in 2D, but to couple the hair generation process with our 3D face model. We build our hair model on a 2D model, MichiGAN [259], which disentangles hair shape, structure, and appearance by processing them separately. Here, shape refers to a 2D binary mask of the hair region, structure is represented as a 2D hair strand orientation map, and appearance refers to the global color and style of the hair, which is encoded in a latent space. We incorporate a pretrained MichiGAN in our training pipeline, which we briefly represent as an encoder-decoder style model in Figure 3.2. When we repose faces at inference time, we couple MichiGAN with our 3D face model to change the shape and structure of the hair without changing its appearance code.

Coupling with Face Model. Our 3D-guided hair manipulation algorithm is illustrated in Figure 3.3. Since our face model reconstructs explicit 3D face shapes, we use these to reason about how the hair will move in 2D by calculating a 2D warp field [266]. We derive the 2D warp field based on the pose-induced movement of the 3D face vertices, then extrapolate the face's warp field to the rest of the image. We use the warp field to warp the hair mask and the hair orientation map in 2D. Since this process can introduce warp artifacts, we regularize the warped masks by projecting them onto a PCA basis calculated from a dataset of binary hair masks of portrait images. The orientation map, on the other hand, is regularized as part of the MichiGAN framework, which outputs a map that is consistent with the warped map and aligned with the regularized hair

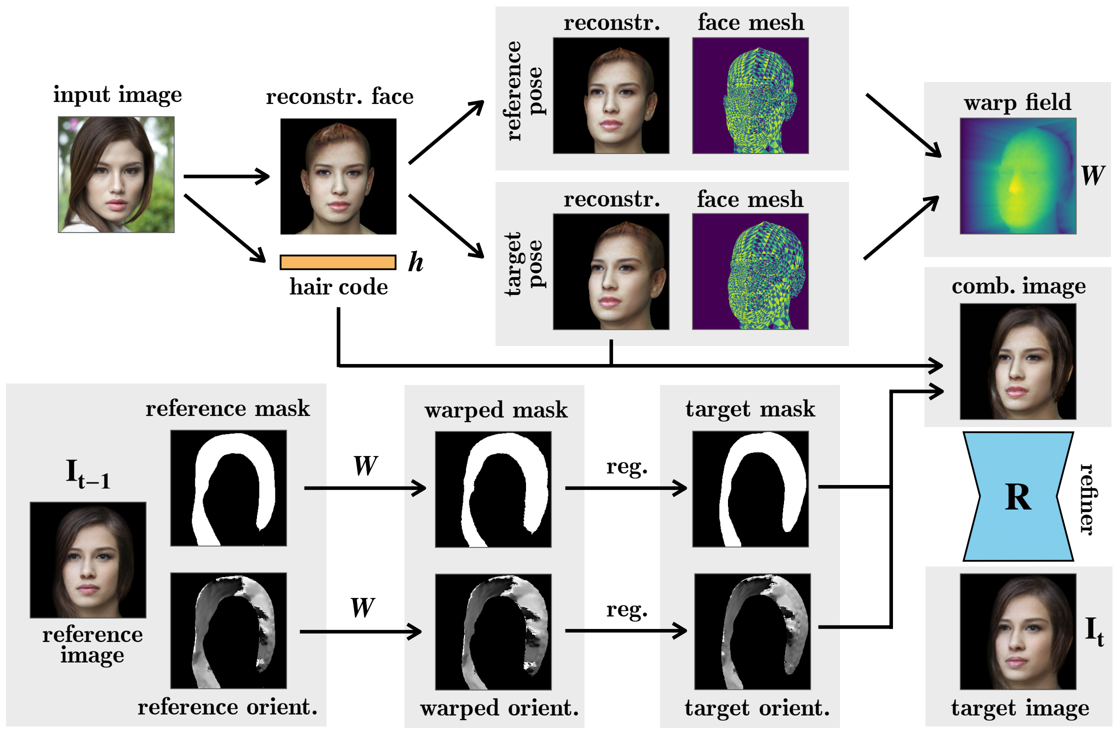

Figure 3.3: **One iteration of our hair manipulation algorithm**. Given a reference pose from the previous iteration and a target pose, we calculate a 2D warp field based on how 3D vertices move within the image plane. Given a reference image from the previous iteration $\mathbf{I}_{t-1}$ along with its reference mask and orientation map, we use this warp field to warp the mask and the orientation map, which we regularize to obtain the target mask and orientation map. Next, we combine these with the hair appearance code obtained from the original input image and the reconstructed face reposed to the target pose, to obtain a novel portrait image $\mathbf{I}_t$. At the end, we feed this image through the refiner to obtain a photorealistic output. This algorithm is invoked sequentially starting from the original pose.

mask. Finally, the reconstructed face in the target pose, hair appearance code, hair mask, and hair orientation map are combined by the MichiGAN pipeline to produce the reposed portrait image, which is then processed with the refiner. For large pose variations, we invoke this algorithm sequentially by going from reference pose to target pose in multiple steps, and we regularize the warped masks and orientation maps at each step.

Warp field calculation. In each iteration of our hair manipulation algorithm, we first identify the visible triangles of the given face mesh with its reference pose and compute the center of each triangle by taking the average of its vertices. Then, we project these triangle centers onto the image plane under both the reference and the target pose. Using the correspondences between the two projections, we construct a 2D warp by calculating how much each of the projected triangle centers moves in pixel space as a result of the pose change. For the vertical component of the warp field, we simply copy the vertical warp component from the nearest neighbor that was assigned a warp. For the horizontal component, we use a heuristic to assign a fixed horizontal warp to every pixel on the left edge of the image and a different fixed horizontal warp to every pixel on the right edge; then the horizontal component of the warp field for the entire image is simply interpolated from the assigned warps. In particular, when we rotate the faces clockwise (counter-clockwise) around the vertical axis, we extend a ray from the center of the 3D face to the left (right) perpendicular to the face's plane of symmetry and identify the 3D point on the ray whose projection lies on the leftmost (rightmost) edge of the image. Next, we calculate by how much this point's projection into the image plane moves when the head rotates, and we multiply this number by 3 to obtain the horizontal warp that we assign to the leftmost (rightmost) column of the warp field. For the rightmost (leftmost) column, we heuristically choose a displacement of 10 pixels to the right (left). Finally, we interpolate between the assigned pixels using linear interpolation to obtain the horizontal warp of every image pixel.

Regularization of the hair mask. After we obtain a complete warp field, we apply it to the reference hair mask and orientation. The orientation is regularized as part of the MichiGAN pipeline, whereas we regularize the mask by projecting it onto a PCA basis that we calculate from a dataset of hair masks. In particular, we construct our hair mask dataset by randomly selecting 10 000 samples from our FFHQ training set and combining it with 10 000 masks that we obtain from the USC Hair Salon database [267]. For the latter, we attach 3D hair models from the USC Hair Salon database to the FLAME mean head model, which we rotate around the vertical axis by an angle uniformly sampled from $[-90°, 90°]$. The hair masks are obtained by rendering these 3D shapes. Finally, after downsampling all masks to 64×64 resolution, we construct a PCA basis with 50 principal components, onto which we project the hair masks at each iteration of our reposing algorithm. In this work, starting from the pose of the original face image, we invoke this algorithm sequentially by imposing a pose change of 5° in each iteration.

3.3.4 Refinement

Although our combined model's rendered 3D reconstructions and 2D hair reconstructions closely resemble the original images, there remains a small realism gap that needs to be filled. In particular, since we regularize the reconstructed albedos using the FLAME albedo space, the reconstructions do not exhibit sufficient variation in the eye regions, and they lack certain details such as eyelashes, facial hair, teeth, and accessories, which are not modeled by the FLAME mesh template. Furthermore, since face and hair are processed separately, some reconstructions have blending issues between the face and the hair. To address these issues, we utilize a refiner network, which closes the realism gap between the reconstructions and the original images while making a minimal change to the reconstructions. We employ a U-Net [268] that takes in a combined image of the reconstructed face and hair and outputs a more realistic portrait image, as shown in Figure 3.2.

After freezing the weights of the rest of the model, we train the refiner with pairs of original images from the dataset and reconstructed images. We combine the identity loss (3.22) described above with an adversarial loss as well as a reconstruction loss based on the VGG-16 perceptual loss [269, 270], promoting better reconstruction quality for hair. Denoting the combined face-and-hair reconstruction as $\hat{\mathbf{x}}_\mathbf{c} := \hat{\mathbf{x}}_\mathbf{f} \odot \mathbf{M}_\mathbf{f} + \hat{\mathbf{x}}_\mathbf{h} \odot \mathbf{M}_\mathbf{h}$, the refined image as $\hat{\mathbf{x}} := \mathbf{R}(\hat{\mathbf{x}}_\mathbf{c})$, and the original face and hair as $\mathbf{x}' := \mathbf{x} \odot (\mathbf{M}_\mathbf{f} + \mathbf{M}_\mathbf{h})$, we employ the following loss function for the refiner:

$$L_{\text{gan}}^{\text{ref}} + 8.0\ \|f_{\text{VGG}}(\mathbf{x}') - f_{\text{VGG}}(\hat{\mathbf{x}})\|_2^2 + 10.0\ \left(1 - \cos(f_{\text{id}}(\mathbf{x}'), f_{\text{id}}(\hat{\mathbf{x}}))\right)\ , \tag{3.34}$$

where the second term is a VGG-16 perceptual loss [269, 270] and the first term $L_{\text{gan}}^{\text{ref}}$ is a GAN loss:

$$L_{\text{gan}}^{\text{ref}} = -\log \mathbf{D}_{\text{ref}}(\mathbf{R}(\hat{\mathbf{x}}_{\mathbf{c}})). \tag{3.35}$$

Here, $\mathbf{D}_{\text{ref}}$ is a discriminator trained to minimize the following loss:

$$L_{\text{disc}}^{\text{ref}} = -\frac{1}{2}\log \mathbf{D}_{\text{ref}}(\mathbf{x}') - \frac{1}{2}\log(1 - \mathbf{D}_{\text{ref}}(\mathbf{R}(\hat{\mathbf{x}}_{\mathbf{c}}))). \tag{3.36}$$

With a batch size of 16, we train the refiner and the discriminator for 500 000 iterations using the Adam optimizer with learning rate 0.0001. To prevent overfitting, we randomly translate $\mathbf{x}'$ and $\hat{\mathbf{x}}_{\mathbf{c}}$ together, where horizontal and vertical translations are uniformly sampled from $[-15, 15]$ pixels.

3.3.5 Architecture Details

Encoders. For encoders $\{\mathbf{E}_{\boldsymbol{\alpha}}, \mathbf{E}_{\boldsymbol{\beta}}, \mathbf{E}_{\boldsymbol{\gamma}}, \mathbf{E}_{\boldsymbol{\theta}}\}$, we employ the ResNet-18 architecture [271] (starting from the ImageNet [272] pretrained weights) where we change the final layer to reflect the dimensionality of each latent representation: $\boldsymbol{\alpha} \in \mathbb{R}^{150}$, $\boldsymbol{\beta} \in \mathbb{R}^{200}$, $\hat{\boldsymbol{\gamma}} \in \mathbb{R}^{27}$, and $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{6}$.

Generators. For the albedo generator $\mathbf{G}_{\boldsymbol{\beta}}$, we employ the original StyleGAN2 [225] architecture up to the 256×256 layer. For the shape generator $\mathbf{G}_{\boldsymbol{\alpha}}$, we use the architecture shown in Table 3.3. The output of this network is a UV-space representation of shape from which we sample points corresponding to the UV-coordinates of each vertex in the FLAME topology [174]. Then, we add these as an offset to the FLAME mean shape to obtain a 3D shape in Euclidean space.

Refiner. We employ a U-Net [268] with 5 convolutional layers followed by 5 transpose convolutional layers with skip connections. We provide the U-Net architecture details in Table 3.4.

3.4 Experiments and Results

In our experiments, we manipulate portrait images with respect to several physical attributes and compare them with the results of a state-of-the-art relighting method [257] and a real-image manipulation method, PIE [20]. Because we generate a full 3D face model, we can manipulate physical attributes beyond the distribution of the training set. We can also modify the face in ways not seen during training, such as relighting using a different lighting and shading model.

3.4.1 Expression and Lighting Manipulation

We illustrate our facial expression and lighting manipulation results in Figure 3.4. To edit facial expression (left), we choose an eigenvector from the FLAME expression basis and multiply it by a constant factor to obtain an offset, which we add to the vertex locations in our model's reconstructed 3D shape. In the moderate examples (top left), we use the first eigenvector to add smile/frown variations up to ±2 standard deviations. In the extreme examples (bottom left), we scale 4 different expression eigenvectors by up to 10 standard deviations. For lighting manipulation (right), the moderate edits (top right) rotate the reconstructed lighting around the camera axis.

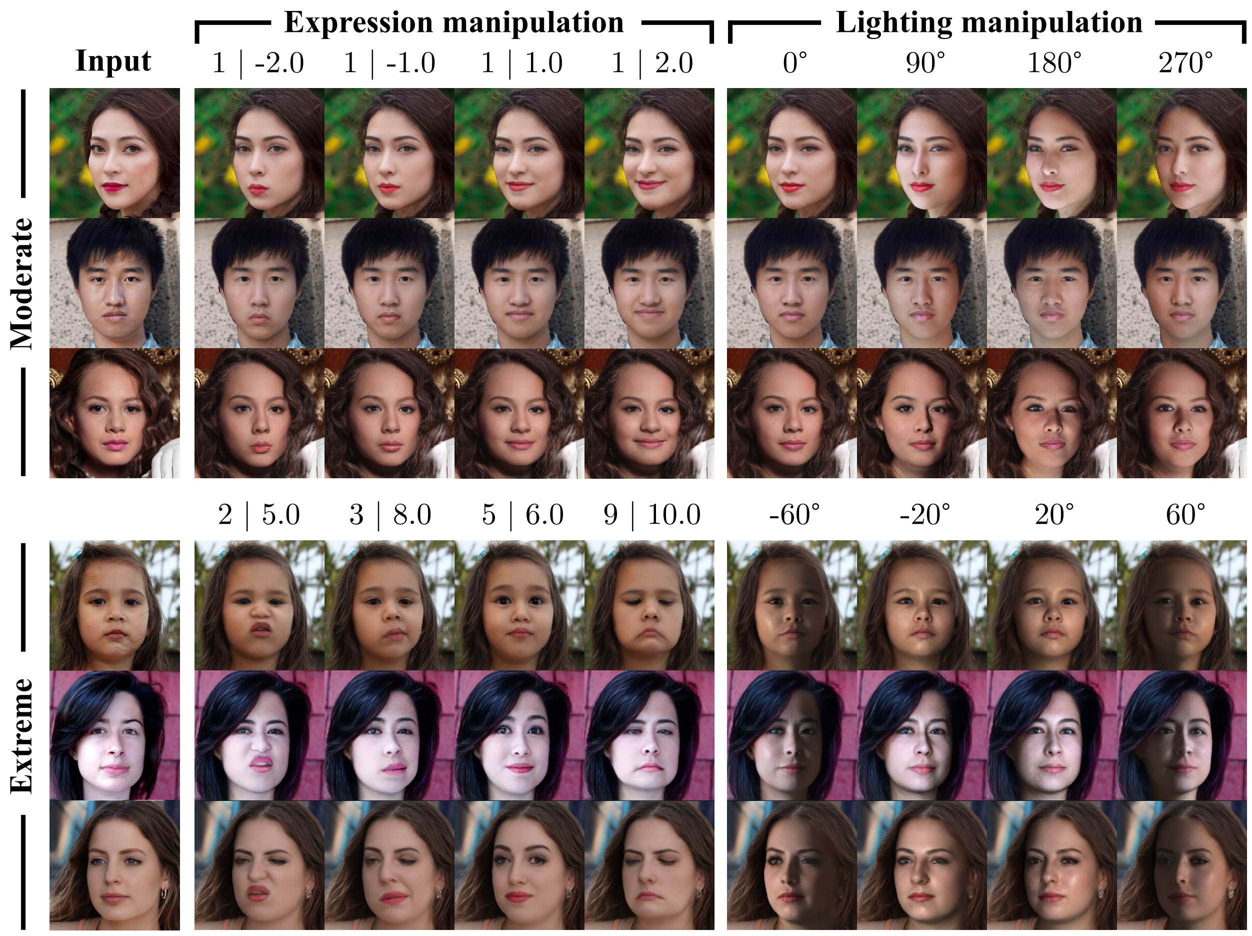

Figure 3.4: **Expression and lighting manipulation results. Expression manipulation (*left*).** We illustrate both moderate (*top*) and extreme (*bottom*) expression variations. The two numbers above each column indicate which FLAME expression eigenvector is used and by how many standard deviations it is scaled. **Lighting manipulation (*right*).** *Top:* For moderate variation, we rotate the reconstructed lighting around the camera axis by the angle above each column. *Bottom:* For extreme lighting variation, we render the reconstructed 3D model using a point light source and Phong shading model.

For extreme lighting variations (bottom right), we employ a point light source and Phong shading model, where we rotate the light source horizontally around the vertical axis and can introduce any desired amount of specularity to the face albedo. The results demonstrate that our method handles extreme expressions and lighting conditions that are not well-represented in the training set and can use lighting and shading models not used in training. Although our method facilitates face image manipulation in several physical attributes simultaneously or in isolation, it is also able to outperform methods that are focused on and optimized for more limited tasks such as manipulating a single attribute. To illustrate this, we compare our extreme lighting manipulation results with those of a state-of-the-art relighting method [257] in Figure 3.5. For additional expression and lighting manipulation results, please refer to Figure 3.10 and Figure 3.11.

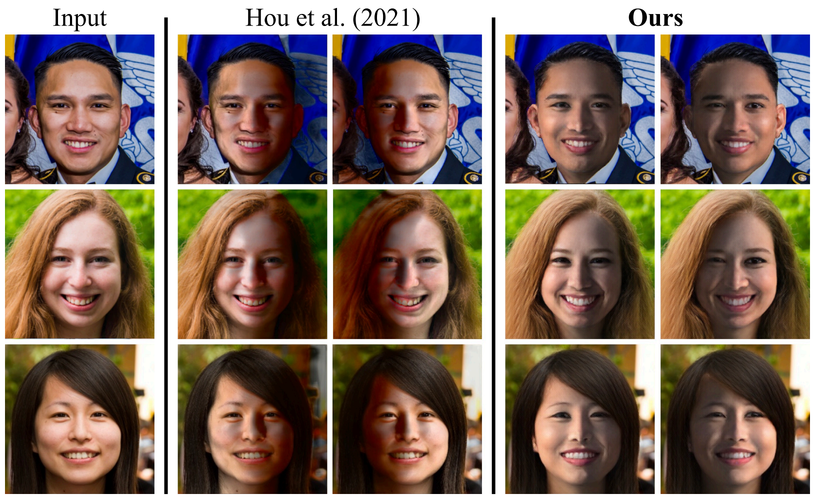

Figure 3.5: **Relighting comparisons**. Relighting comparison with Hou et al. [257], where we use a point light source. Our method achieves more photorealistic relighting with fewer artifacts.

3.4.2 Shape Transfer

Our model achieves better disentanglement of physical attributes such as shape and albedo by modeling them separately and explicitly. This disentanglement is illustrated by the shape transfer results in Figure 3.6, where we transfer the 3D face shape of a source image to a target image. Our method (*left*) is able to transfer the face shapes accurately, while maintaining photorealism and keeping the albedo, lighting, and hair unchanged. This is in contrast to the shape transfer results by the previous state-of-the-art (*right*, a combination of StyleRig [19] and PIE [20]), where for a given source shape, the transfer results have varying face shapes with noticeable differences in expressions. When the source and target images are identical (images on the diagonal), our method produces the original reconstruction by design, whereas PIE + StyleRig struggles to maintain the original identity. For additional shape transfer results, please refer to Figure 3.12.

3.4.3 Joint Transfer of Physical Attributes

The full disentanglement of our face model is demonstrated by its ability to transfer all physical attributes either individually or jointly. In Figure 3.7, we illustrate our results for joint albedo and lighting transfer, as well as joint transfer of albedo, lighting, and shape.

3.4.4 Interpolation in the Latent Space

Although we do not impose any smoothness constraints within the latent spaces, the learned shape and albedo latent spaces enable smooth interpolation between different latent codes. We present our interpolation results in Figure 3.8, where we simultaneously interpolate between the reconstructed shape code, albedo code, and lighting parameters of the reference and target images.

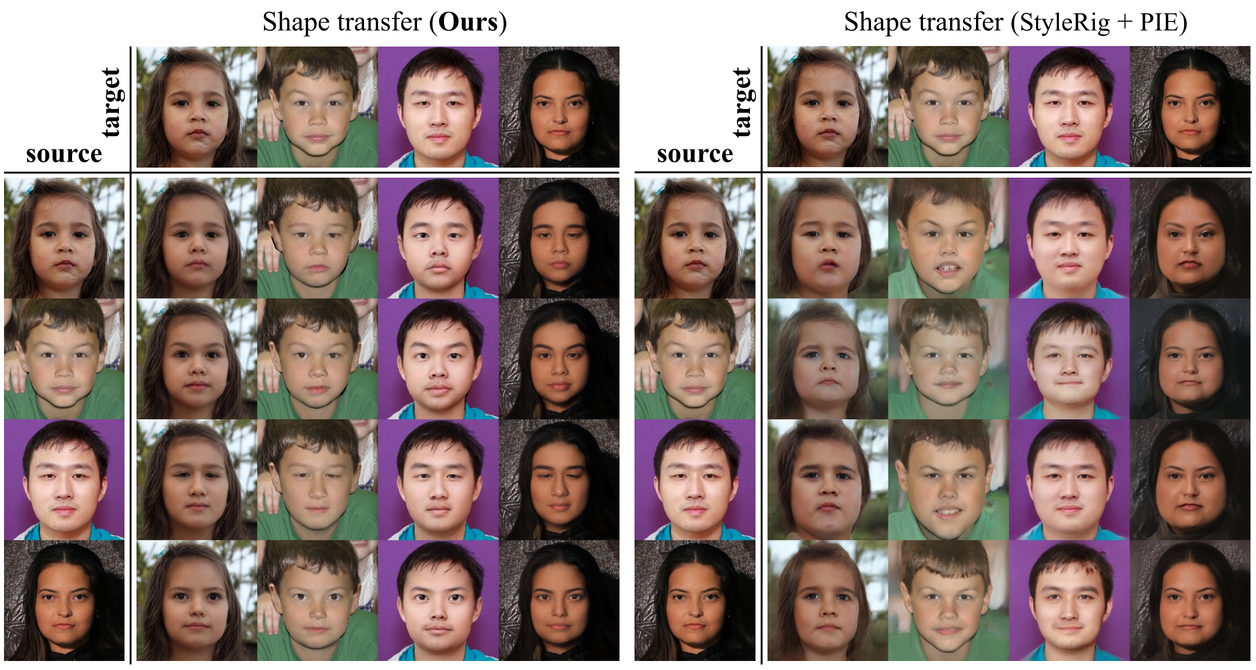

Figure 3.6: **Shape transfer comparisons.** We transfer the 3D shape of each source image to each target image while keeping everything else unchanged. Our results (*left*) demonstrate more accurate shape transfer and much better disentanglement between shape and other attributes (e.g., albedo, pose, and hair) than the combination of StyleRig [19] and PIE [20] (*right*).

3.5 Ablation Study

Training and loss functions. In our experiments, we observe that pretraining on synthetic data is crucial for our method to work, since we observe stability issues when we started out by training on real data. For the real data training, our experiments suggest that all loss functions except for the albedo regularization, lighting regularization, and regularization of shape and albedo codes are crucial for our method to achieve reasonable face reconstructions. When we omit albedo and lighting regularizations, we observe that our method converges to a state in which the albedo reconstructions are washed out and the appearance of the face is mostly attributed to the lighting, which suggests that albedo and lighting regularizations are important to achieve better albedo and lighting disentanglement.

Refinement. In this section, we analyze the impact of the refiner on our reconstructions by providing a qualitative and quantitative comparison of our results with vs. without the refiner. In addition, we compare our reconstructions with a state-of-the-art nonlinear 3D morphable model proposed by Tran and Liu [178]. We illustrate our qualitative comparisons in Figure 3.9, where our reconstructions achieve better photorealism than those of Tran and Liu. We also observe a notable improvement in photorealism using our complete model (with refiner) vs. using our model without the refiner. To quantify our observations, we calculate a face recognition (FR) score as the average cosine similarity between the feature vectors extracted from the ArcFace face recognition network [264] for the original and reconstructed images (from our test dataset).

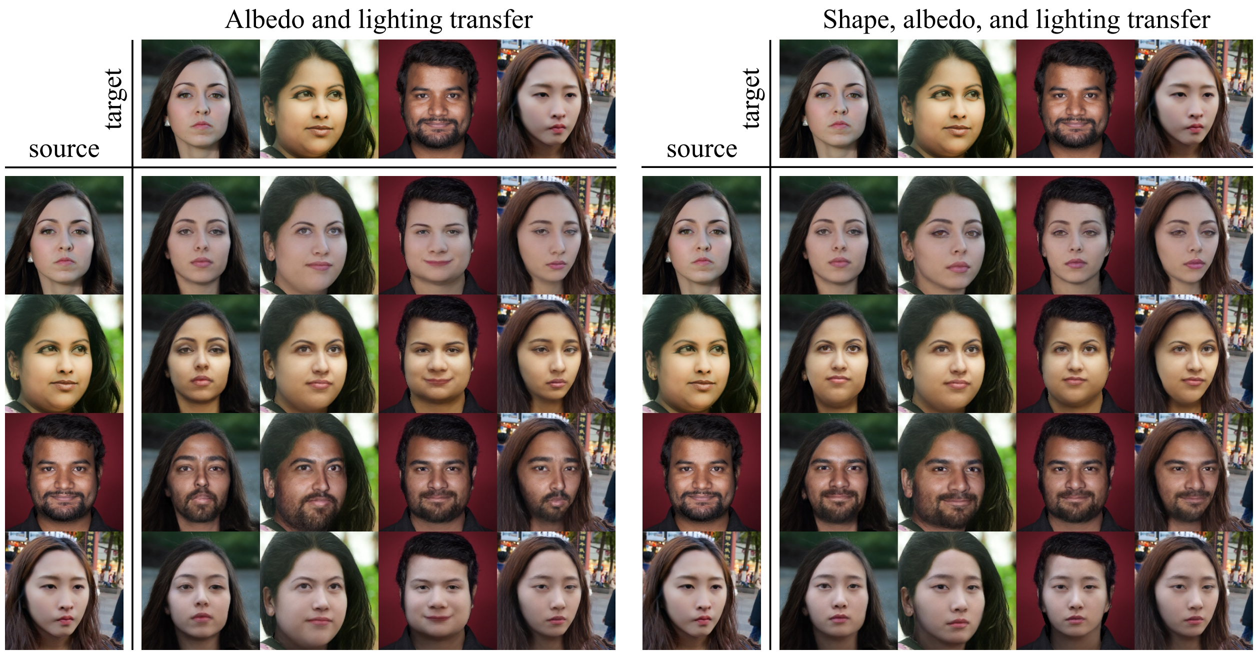
Albedo and lighting transfer
target
source
Shape, albedo, and lighting transfer
target
source

Figure 3.7: **Joint transfer of physical attributes**. Our method is able to transfer physical attributes jointly as well as individually. In this figure, we jointly transfer the indicated physical attributes of each source image to each target image while keeping the other parameters of the target image unchanged. *Left:* Albedo and lighting transfer. *Right:* Shape, albedo, and lighting transfer.

Table 3.1: Average face recognition (FR) scores, SSIM, PSNR, and LPIPS between the original and reconstructed face images from our dataset. We observe a notable improvement due to the refinement.

	FR score ↑	SSIM ↑	PSNR ↑	LPIPS ↓
Ours (without refinement)	0.66 ± 0.10	0.72 ± 0.09	22.24 ± 2.63	0.15 ± 0.05
Ours (with refinement)	**0.68** ± **0.10**	**0.74** ± **0.08**	**23.20** ± **2.47**	**0.12** ± **0.04**

We also compute the structural similarity index measure (SSIM) [273], peak signal-to-noise ratio (PSNR), and learned perceptual image patch similarity (LPIPS) [274] between the original and reconstructed images. We quanitatively compare our model with versus without the refiner in Table 3.1. In Table 3.2, we perform quantitative comparisons. To obtain the results in Table 3.2, we masked out the hair, background, clothing, and teeth for fair comparison.

3.6 Discussion and Outlook

In this chapter, we introduced a novel framework for performing face image manipulation in a 3D-aware manner. We demonstrated that a fully 3D architecture, designed to process facial attributes independently, enables fast, disentangled, and photorealistic manipulations. Our approach integrates the physically-grounded representation of 3DMMs with the expressive capacity of style-based generators, implemented within an encoder–decoder architecture built on the 3DMM template. In contrast to prior methods that rely on iterative optimization procedures to

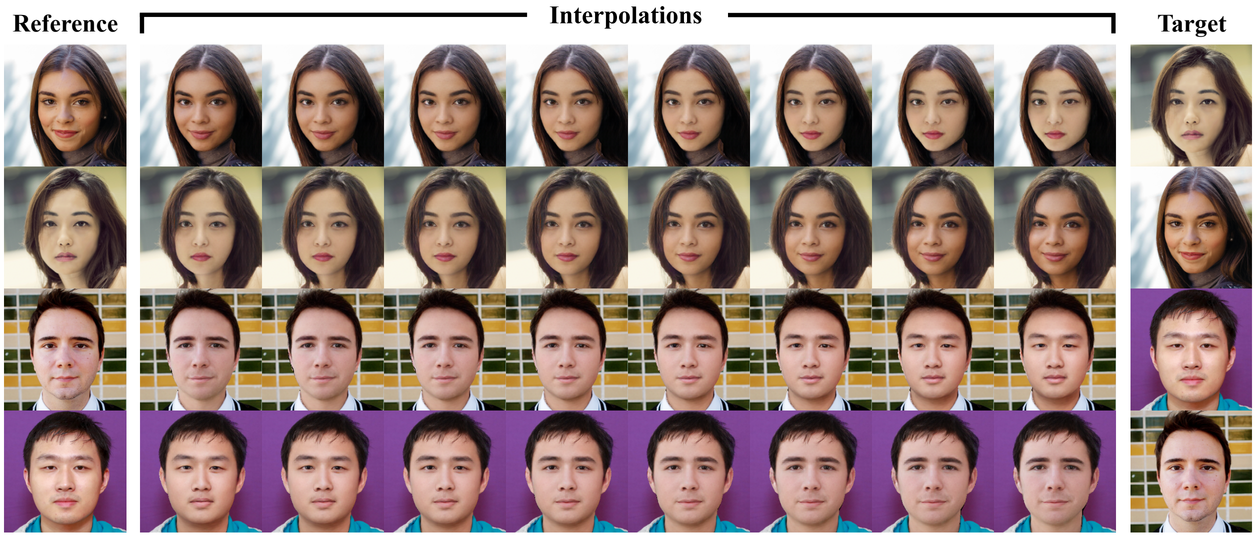

Figure 3.8: **Interpolation results.** In each row, given a reference and a target image, we interpolate between their shape codes, albedo codes, and lighting parameters. For each interpolated image, the background and the latent code for hair are copied from the reference image.

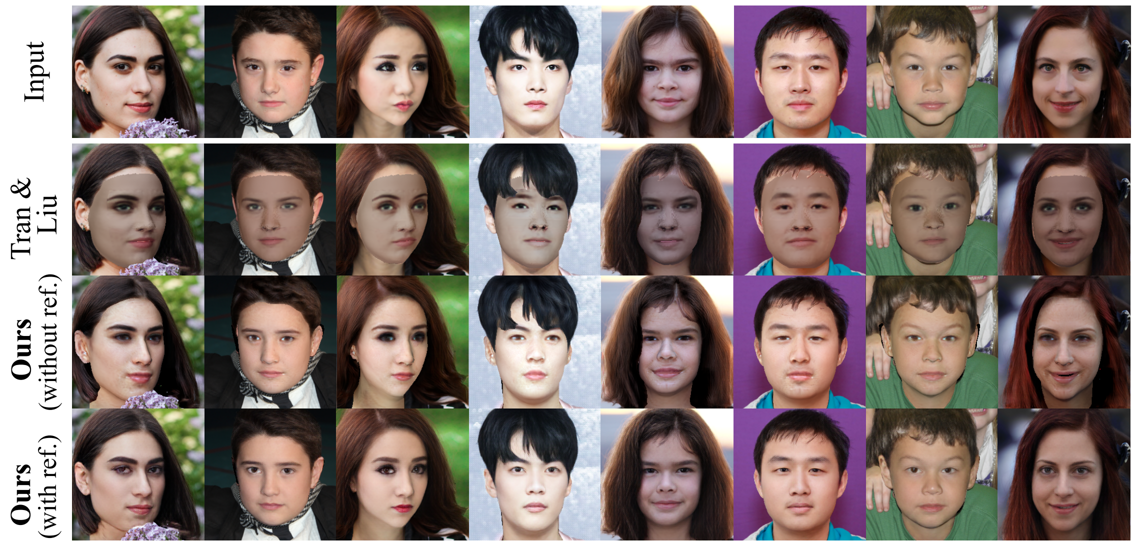

Figure 3.9: **Ablation study and comparison with Tran and Liu** [178]. On images from the test set (never seen during training), our method is able to reconstruct faces more accurately and photorealistically than [178]. These results also demonstrate that our full model (with refinement) shows a notable improvement in image quality vs. our model without the refiner.

manipulate real images, our encoding mechanism supports efficient, feed-forward manipulation at inference time. A key insight from our design is that operating on attributes in their physical spaces facilitates generalization beyond the distribution of the training data. Empirical results validate the ability of our method to manipulate 3D shape, albedo, and lighting photorealistically, producing larger variations compared to prior work while achieving better disentanglement.

Table 3.2: Average face recognition (FR) scores, SSIM, PSNR, and LPIPS between the original and reconstructed face images for our method (both with and without refinement) and Tran and Liu [178]. To obtain the results in this table, we masked out the hair, background, clothing, and teeth for fair comparison with Tran and Liu [178]. Our method achieves better scores in all three metrics.

	FR score ↑	SSIM ↑	PSNR ↑	LPIPS ↓
Tran and Liu [178]	0.51 ± 0.12	0.87 ± 0.03	20.96 ± 1.57	0.092 ± 0.026
Ours (without refinement)	0.69 ± 0.10	0.87 ± 0.04	25.08 ± 2.92	0.086 ± 0.035
Ours (with refinement)	**0.71** ± **0.10**	**0.88** ± **0.04**	**26.17** ± **2.71**	**0.062** ± **0.031**

Limitations. While our methodology successfully addresses the core motivations outlined at the beginning of this chapter, it exhibits several limitations that may need to be resolved for certain applications. One notable limitation in our pipeline design is the explicit disentanglement of hair from the physical attributes of the face. As a result, changes to other attributes, such as illumination, do not affect the appearance of hair. Although our refinement stage adjusts hair appearances to enhance visual realism, the resulting changes are often subtle and insufficient for high-quality portrait relighting tasks. This limitation could potentially be mitigated by coupling face attributes with the hair model at training time. Additionally, we observe that our model tends to attribute variations in skin tone primarily to lighting rather than albedo. This bias arises from our use of the FLAME albedo basis to regularize albedo predictions, as the basis itself lacks diversity in skin tones. Adopting a more representative and demographically balanced albedo model that is better aligned with global population statistics could lead to more accurate albedo estimates and, consequently, more realistic relighting under novel illumination conditions. Finally, because our 3D shape predictions are generated via a convolutional architecture that enforces local smoothness across neighboring mesh vertices, our model often produces overly smooth geometry. As a result, fine-scale face details, such as wrinkles, are sometimes encoded into the albedo rather than the geometry. This misattribution has a direct impact on relighting quality, as physically plausible lighting interactions with high-frequency surface features like wrinkles are critical for photorealistic rendering of faces.

The next era. While the work presented in this chapter offers valuable insights into the design of a 3D-aware face manipulation pipeline under constraints imposed by specific data modalities and desiderata such as inference speed and generalization, the broader landscape of the field has evolved significantly at the time of this writing. In particular, adversarial training frameworks [84, 205]—once dominant despite their well-known instability—have been largely supplanted by diffusion-based models [89, 275]. These newer paradigms offer more stable training dynamics and have demonstrated the ability to model significantly more complex, high-dimensional data distributions, including video and 3D assets [276–278]. Today, considerable industrial investment is directed toward training large-scale variants of these diffusion models on massive datasets, with some efforts resulting in open-source releases that enable further research [279, 280]. Recent progress has shown that these models can be fine-tuned for specific downstream tasks or utilized as sources of distilled knowledge to supervise the training of task-specific models [281, 282]. This has accelerated the research and development cycle, reducing the overall effort required to build high-performing systems compared to earlier approaches. Beyond the pure image-based diffusion models, the availability of large-scale visual–textual data pairs has enabled the emergence of text-conditioned visual content generation [283], where natural language prompts serve as intuitive control signals for a wide array of synthesis tasks. This modality has opened up new possibilities for human–AI interaction, allowing for flexible and expressive manipulation of visual content through simple textual descriptions [284, 285].

In addition to the generative pipelines led by diffusion models, transformer-based architectures [286], which underpin large language models (LLMs), have recently revolutionized the machine learning landscape, demonstrating capabilities that closely resemble human-level intelligence in natural language understanding and generation [287]. These models have since been extended to multimodal settings, enabling interaction with data across various modalities such as images, audio, and video [288]. When paired with visual inputs, they are often referred to as vision-language models (VLMs) [289, 290]. Such models can serve as a rich source of distilled knowledge, making the implicit representations they encode more accessible for downstream tasks. In the context of face image manipulation, these models offer a substantial paradigm shift. Rather than constructing pipelines from the ground up—by carefully designing network architectures, curating datasets, engineering loss functions, and managing inference-time considerations—the main objectives shift toward effectively leveraging the hidden knowledge embedded in these pretrained models. Such reframing enables creative repurposing of this hidden knowledge to solve complex visual tasks with minimal supervision. Indeed, this emerging workflow is rapidly overtaking traditional approaches, as the performance gains achieved through large-scale pretrained models are often unprecedented and difficult to match through conventional means [291–293].

Table 3.3: **Architecture of the shape generator $\mathbf{G}_\alpha$**. The output of the network is a UV-representation of 3D shape, where the three channels of the 256×256 output represent 3D offsets (in x, y, and z) from the FLAME mean head shape [174].

layer type	**kernel size / stride**	**output shape**	**activation**
linear	–	1024×1	none
reshape	–	$16 \times 8 \times 8$	–
conv2d and upsample	4×4 / 1	$32 \times 16 \times 16$	tanh
conv2d and upsample	4×4 / 1	$64 \times 32 \times 32$	tanh
conv2d and upsample	4×4 / 1	$64 \times 64 \times 64$	tanh
conv2d and upsample	4×4 / 1	$64 \times 128 \times 128$	tanh
conv2d and upsample	4×4 / 1	$64 \times 256 \times 256$	tanh
conv2d	4×4 / 1	$3 \times 256 \times 256$	tanh

Table 3.4: **Architecture of the refiner R**. We employ a U-Net [268] with skip connections between the encoder and decoder parts of the network. In all layers of the encoder, we use the LeakyReLU activation function with a negative slope of 0.2. All layers of the decoder use the ReLU activation function.

layer type	**kernel size / stride**	**output shape**	**activation**
conv2d	4×4 / 2	$64 \times 128 \times 128$	LeakyReLU
conv2d	4×4 / 2	$128 \times 64 \times 64$	LeakyReLU
conv2d	4×4 / 2	$256 \times 32 \times 32$	LeakyReLU
conv2d	4×4 / 2	$512 \times 16 \times 16$	LeakyReLU
conv2d	4×4 / 2	$512 \times 8 \times 8$	LeakyReLU
conv2d_transpose	4×4 / 2	$512 \times 16 \times 16$	ReLU
conv2d_transpose	4×4 / 2	$256 \times 32 \times 32$	ReLU
conv2d_transpose	4×4 / 2	$128 \times 64 \times 64$	ReLU
conv2d_transpose	4×4 / 2	$64 \times 128 \times 128$	ReLU
conv2d_transpose	4×4 / 2	$3 \times 256 \times 256$	ReLU

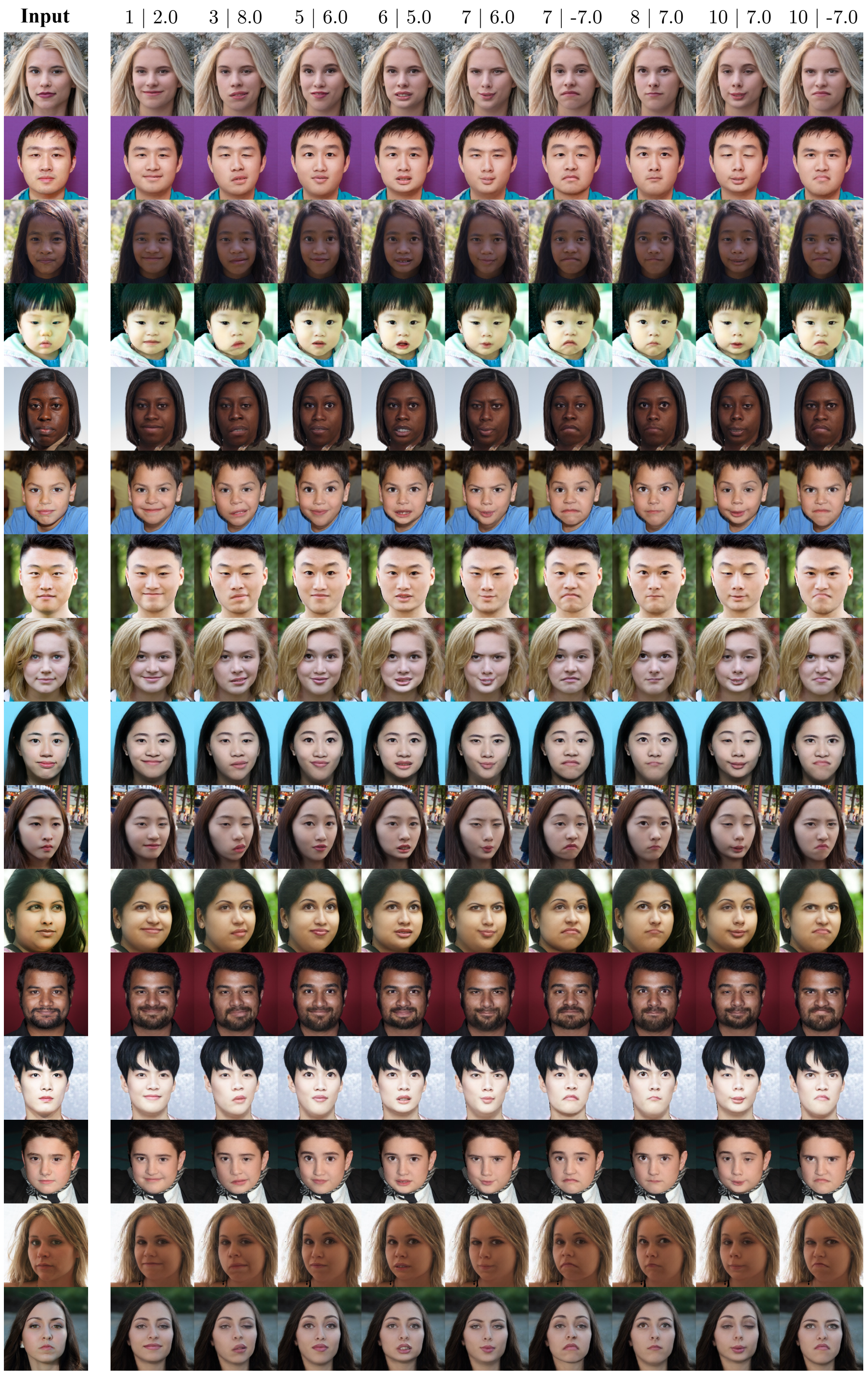

Figure 3.10: **Additional expression manipulation results**. We illustrate several expression changes with varying intensities. The two numbers above each column indicate which FLAME expression eigenvector is used and by how many standard deviations it is scaled.

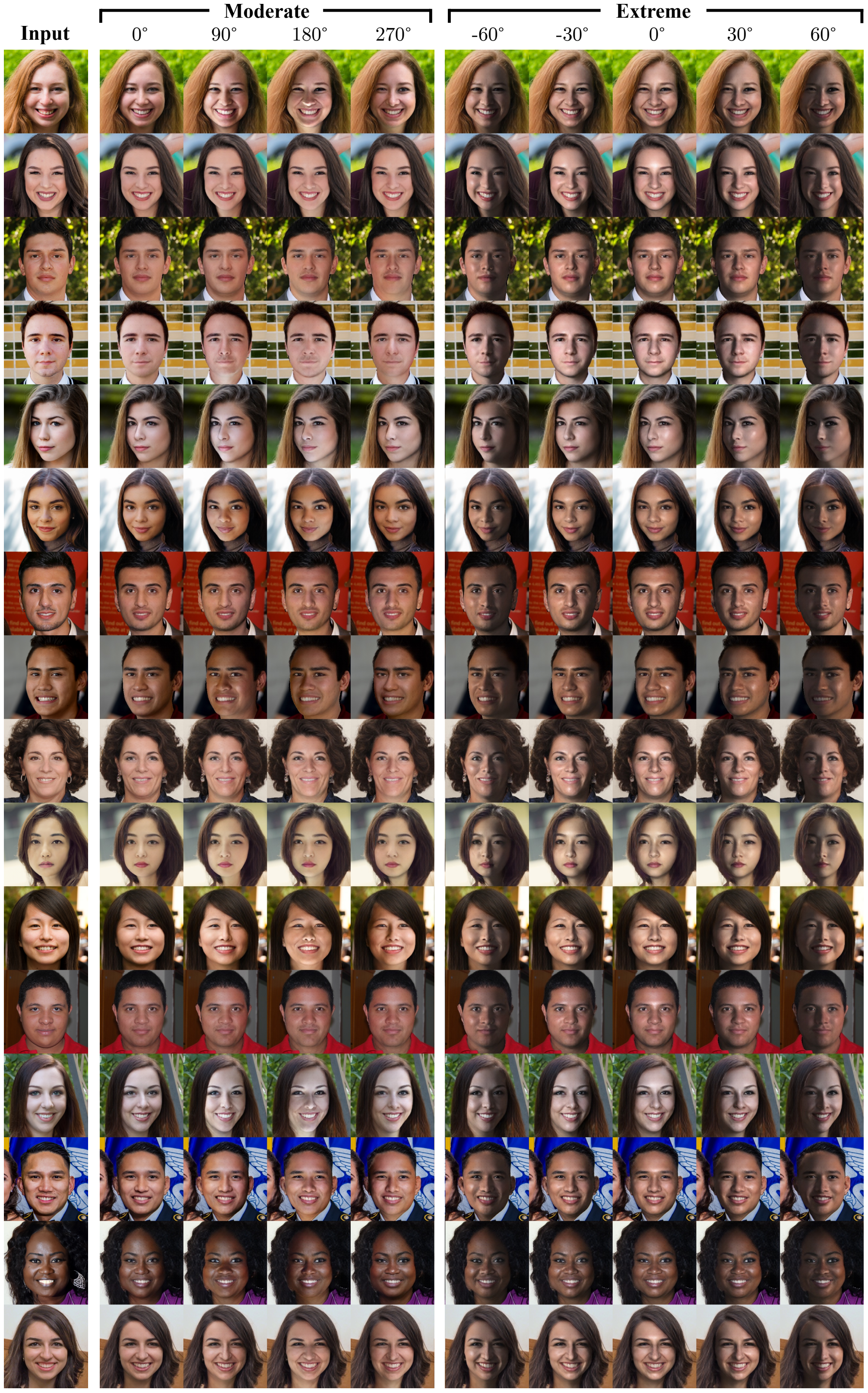

Figure 3.11: **Additional lighting manipulation results.** For moderate variation, we rotate the reconstructed lighting around the camera axis by the angle listed above each column. For extreme lighting, we render the reconstructed 3D model using a point light source and Phong shading model.

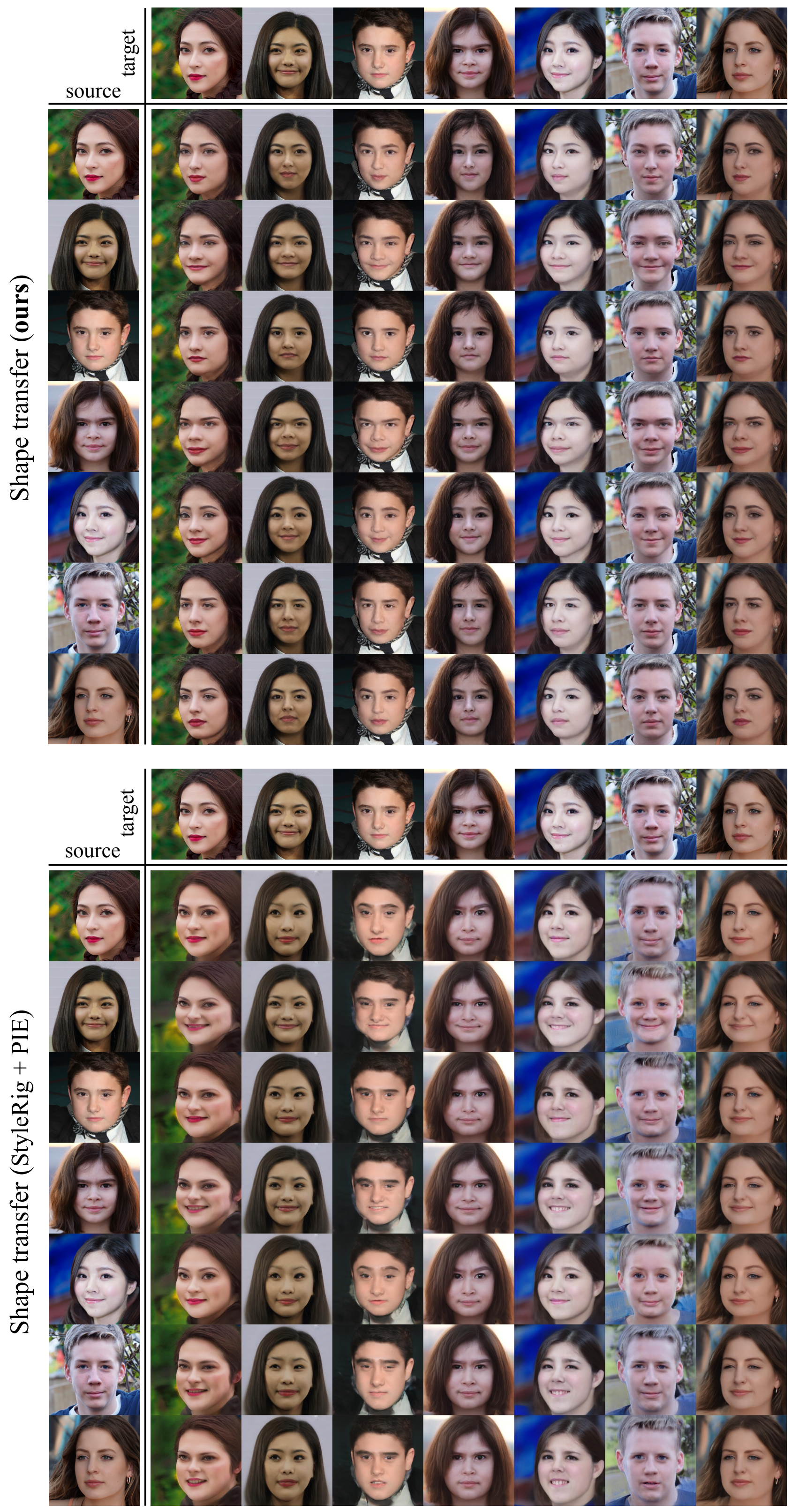

Figure 3.12: **Additional shape transfer results and comparisons**. We transfer the 3D shape of each source image to each target image while keeping everything else unchanged. Compared to the previous state of the art (StyleRig [19] + PIE [20]), our results (*top*) demonstrate more accurate shape transfer and much better disentanglement between shape and other face attributes (e.g., albedo, pose, and hair).

4 Efficient Rendering of Dynamic Faces

4.1 Introduction

From this point onward, we narrow our focus to the domain of digital twins, with a particular emphasis on applications in extended reality (XR). Recent advances in computer vision and graphics, coupled with significant progress in mobile and embedded hardware, have brought the synthesis, editing, and animation of digital twins on consumer-grade devices from concept to reality [294]. Yet, this reality remains in its early stages of maturity. Edge devices such as head-mounted displays are still far from ubiquitous, hindered by factors such as high production costs, societal hesitance toward wearable technology, as well as the limited readiness and efficiency of current XR applications for broad adoption [295]. In the remainder of this thesis, we examine the underlying causes of these challenges in efficiency and fidelity, and identify the key technical considerations. Then, we develop frameworks that address these key considerations to enable more accessible and scalable deployment of digital twin technologies in everyday XR systems.

While the synthesis of 3D faces from 2D image data once again serves as the foundation of our development, our focus in this chapter shifts away from multi-attribute editing. Instead, our objective is to render temporally coherent sequences of facial expressions that simulate lifelike conversations through virtual digital twins within XR environments. Here, we explore two application scenarios. First, by relaxing the requirements on controllability and drivability, we aim to synthesize 3D facial expression sequences that can serve as 3D memories: pre-recorded, expressive animations that can be viewed on a mobile device or experienced immersively through a head-mounted display. Second, we consider the more challenging case of real-time controllability, where a tracked sequence of facial expressions is used to drive a 3D face representation, enabling virtual telepresence where users interact with each other in real-time. These responsive, interactive digital twins are commonly referred to as *avatars*.

Controllability of 3D face representations introduces significant challenges, particularly in terms of generalizing to unseen expressions or novel facial dynamics beyond the training distribution. For this reason, the current chapter focuses on the first, more constrained problem. To lay the groundwork for both this chapter and the next, we begin by reviewing the core challenges shared across both application settings. When we turn our full attention to controllable avatars in the following chapter, we also address the additional difficulties introduced by real-time driving and expression generalization.

4.1.1 Main Challenges

Capturing and simulating human presence in 3D digital form is not a recent concept. Early examples appeared in video games and cinematic productions as far back as the 1990s, marking the rise of computer-generated imagery (CGI) [296, 297]. Thanks to major advances in graphics hardware and software, and particularly in performance capture technologies, the production costs of CGI have decreased significantly, enabling increasingly realistic depictions of digital humans in the entertainment and communications industry [298, 299]. Beyond entertainment, industries such as advertising, customer service, and healthcare have also begun to embrace digital humans as part of their evolving technological landscape. With the advent of consumer-grade virtual reality, digital humans have found a new medium, giving rise to the concept of 3D digital twins, primarily for applications like telepresence [31]. These digital twins of real humans, referred to as avatars, have attracted considerable attention over the past few years, with many works focusing on the synthesis, manipulation, and animation of photorealistic avatars [134].

Avatars are commonly modeled using either full-body or head-only representations. This thesis focuses exclusively on head-only avatars, thereby avoiding the additional complexities associated with full-body modeling (such as clothing simulation, garment animation, and full-body motion capture) that pose substantial technical challenges beyond those encountered in face modeling [300]. From a production and wide-scale deployment perspective, avatar representations must be evaluated along several key aspects. We categorize these considerations as 1) photorealism, 2) rendering and memory efficiency, and 3) other practical factors.

Photorealism. The photorealism of an avatar is primarily governed by the capacity of the underlying representation. This encompasses not only the static visual fidelity of the reconstructed face but also the level of realism with which dynamic expressions are rendered. Traditionally, mesh-based representations have dominated the digital human landscape [119, 120]. Nowadays, modern mesh reconstruction techniques can achieve high-resolution detail, capturing fine geometric features such as wrinkles, dimples, and skin folds, while also enabling efficient animation or expression control through well-established algorithms [244]. However, as mentioned earlier, meshes are inherently constrained in their ability to model complex volumetric and view-dependent structures. Volumetric approaches such as NeRFs [49] and 3DGS [50], along with their dynamic extensions, offer significantly richer expressiveness. These methods can capture subtle facial idiosyncrasies and specularities that are difficult to reproduce with single surfaces, and can achieve remarkable realism in both static and animated settings [40, 139, 156, 301]. Despite this, they may suffer from visual artifacts, such as *floaters* [302] or *pops* [303], which can detract from immersive experiences. Nevertheless, ongoing research continues to address these limitations, and the state-of-the-art has now reached a level of hyperrealistic synthesis and animation that is increasingly viable for real-world applications.

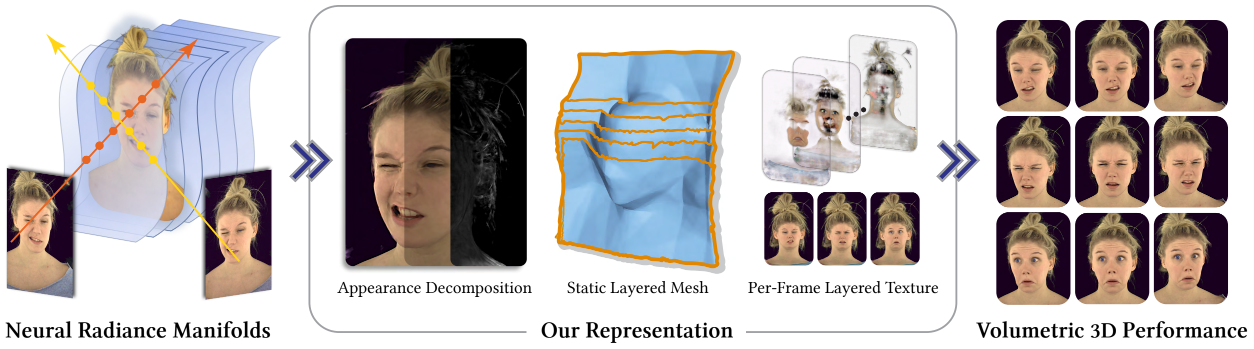
Appearance Decomposition
Static Layered Mesh
Per-Frame Layered Texture
Neural Radiance Manifolds
Our Representation
Volumetric 3D Performance

Figure 4.1: We model a dynamic face sequence as a set of radiance manifolds, which are exported as a static layered mesh and an animated texture. This allows for smoothly controlling the quality vs. memory and compute footprints, while achieving efficient and photorealistic rendering of volumetric scenes using established graphics pipelines without any neural network integration.

Rendering and Memory Efficiency. Classical head-avatar pipelines represent geometry as a low-vertex mesh with a handful of blendshapes. While these meshes fit easily into memory and can be rasterized at lightning speed, their baked textures and simple bidirectional reflectance distribution functions (BRDFs) rarely cross the *uncanny valley* [304]. Volumetric radiance fields—first NeRFs [49], then 3DGS [50]—trade kilobytes of triangle primitives for tens to hundreds of megabytes of density and color, and rely on techniques such as long ray-marches or splat compositing. This exposes the *rendering efficiency vs. memory efficiency* frontier: NeRFs sit on the slow and lightweight end (small MLP, many samples per ray), whereas 3DGS on the fast and heavy end (efficient splat look-up, but potentially millions of splats). Recent work has aimed to push the frontier outward for *general* scenes with techniques such as pruning low-contribution Gaussians at run time [305] or quantizing spherical harmonics coefficients and covariances to lower bit precision [159]. Replicating these tricks in the 3D face domain would essentially exploit the non-uniform signal spectrum of human heads, which manifests high detail around eyes, hairlines, and mouth but is mostly smooth elsewhere. Therefore, pushing the frontier would be primarily achieved by spending bytes and flops where the viewer is paying attention the most.

Other considerations. At production scale, a head avatar pipeline is a relay race that begins with lightning-fast on-device face tracking systems [306]. The tracked coefficients then drive a mesh, NeRF, or 3D Gaussian cloud in real-time. When the users are remote, an avatar system also requires careful determination of what to stream and how to stream it, as there is no established coding standard for transmitting custom 3D assets, unlike the H.264/H.265 standard used for video streaming. Due to the increasing popularity of 3DGS-based representations, this problem has recently sparked new algorithms for 3DGS compression and streaming [157–160]. Furthermore, since companies develop their own ecosystems with custom rendering pipelines and streaming algorithms, cross-device interaction between consumers remains a challenging problem.

4.1.2 Main Objective and Contributions

In this chapter, given observations of a dynamic 3D face, our objective is to learn 1) 3D geometry and 2) appearance *sequences* that produce photorealistic images, support high frame rates, and maintain a modest memory footprint, all while remaining compatible with legacy graphics pipelines and existing streaming infrastructure. To meet these requirements, we propose a novel representation that models 3D faces using a static layered mesh coupled with an RGBA texture video. Our approach is guided by the following key insights:

- Volumetric effects can be effectively approximated using a moderate number of layered surface representations, eliminating the need for fully volumetric models that define geometry and appearance at every point in 3D space.
- Meshes offer efficient rasterization on standard graphics platforms, ensuring backward compatibility and a favorable memory-compute tradeoff.
- Texture videos can be trivially streamed using existing infrastructure, leveraging mature video codec systems widely deployed in today's content delivery networks.

To learn this layered mesh and dynamic texture representation, we build on recent advances in neural rendering, particularly radiance manifolds [307], which enable us to discretize the 3D scene into a set of continuous 2D surfaces. These surfaces are subsequently discretized into textured meshes, which are deployed to game engines on consumer devices. Our system achieves photorealistic and real-time playback of 3D face sequences with challenging geometry and appearance changes. Please refer to Figure 4.1 for an overview of our pipeline.

4.1.3 Preliminaries

In this chapter, we develop a framework for rendering 3D face sequences, where we do not yet focus on the animation problem. Formally, we consider a variation of the model in (2.25):

$$\mathbf{y}_{vt} = \mathcal{R}(G_t, A_t; \mathbf{c}_v) + \mathbf{n} \tag{4.1}$$

Let $\mathcal{Y} \triangleq \{\mathbf{y}_{vt} \mid v = 1, \dots, V;\ t = 1, \dots, T\}$ denote a multi-view video sequence of a *single subject*, where T is the number of frames and V is the number of views. Our objective is to estimate a *sequence* of geometry and appearance realizations $\{G_t\}_{t=1}^{T}$ and $\{A_t\}_{t=1}^{T}$ from our observations $\mathcal{Y}$ as well as the parameters of cameras from each view $\{\mathbf{c}_v\}_{v=1}^{V}$. Since we assume that the rendering operation is deterministic and known, we have a non-blind inverse problem with a non-linear forward model. Still, since there are infinitely many realizations of geometry and appearance sequences that satisfy our observations, this inverse problem is also an ill-posed one. In the next sections, we use a notation consistent with the published work [308].

4.2 Background and Related Work

In this section, we first review dynamic neural radiance fields, providing a brief survey of the core design principles of prominent methods in the literature. We then narrow our scope to generative radiance fields, with an emphasis on computational challenges associated with high-fidelity 3D synthesis. Finally, we limit our attention to a specific generative method, *generative radiance manifolds* [307], whose underlying representation plays a central role in the volumetric representation we develop later in this chapter.

4.2.1 Dynamic Neural Radiance Fields

The advent of Neural Radiance Fields (NeRFs) [49] has had a transformative impact on the computer vision community, catalyzing many lines of research on continuous volumetric representations for photorealistic 3D scene reconstruction and view synthesis. While the original NeRF formulation was designed for static scenes, subsequent efforts have extended it to dynamic environments, where the input consists of monocular or multi-view videos capturing temporally varying 3D content [40, 41, 140, 309, 310]. A natural extension to accommodate dynamics is to append time as an additional input to the radiance field, resulting in a formulation $F_\Theta : (\mathbf{x}, \mathbf{d}, t) \mapsto (\mathbf{c}, \sigma)$, where $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{d} \in \mathbb{R}^2$ respectively denote the spatial location and the viewing direction, and $t \in \mathbb{R}$ the time coordinate. This baseline approach was formalized in Li et al. [140], where the time is encoded using positional encoding similar to the spatial coordinates.

For dynamic scenes, spatial and temporal variations exhibit distinct statistics and frequency characteristics, and treating time as an additional spatial-like coordinate constrains the ability to represent complex and often non-smooth deformations that characterize real-world dynamic scenes [140]. As a result, a single global network struggles to accurately reconstruct detailed dynamics such as facial expressions, fluid motions, or events like occlusions, disocclusions, or disappearing surfaces. Moreover, phenomena involving volumetric discontinuities—e.g., flames, smoke, or splashes—violate the smoothness assumptions baked into (finite-frequency) positional encoding, leading to artifacts and training instability. To address these limitations, Li et al. [140] propose augmenting the NeRF with per-frame learned latent codes $\mathbf{z}_t$ instead, as

$$F_\Theta : (\mathbf{x}, \mathbf{d}, \mathbf{z}_t) \mapsto (\mathbf{c}, \sigma) \,. \tag{4.2}$$

Such a formulation introduces additional expressivity to the model, enabling the network to better capture frame-specific variations without requiring an explicit physical model of motion.

An alternative formulation for modeling dynamic scenes involves decomposing the radiance field into a *canonical* scene representation and a time-dependent deformation field that warps points from the observation space into a shared canonical space, allowing the network to disentangle temporal variations from spatial content [309]. Specifically, for a 3D point $\mathbf{x}$ at time t, we can decompose a dynamic radiance field as follows:

$$G_\Psi : (\mathbf{x}, t) \mapsto \delta\mathbf{x} \qquad F_\Theta : (\mathbf{x} + \delta\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma) \,, \tag{4.3}$$

where G_Ψ is a time-conditioned deformation network, parameterized as an MLP with weights $\boldsymbol{\Psi}$, and F_Θ denotes a static radiance field in canonical space, which is queried by deformed points $\mathbf{x} + \delta\mathbf{x}$. Compared to latent-conditioned dynamic NeRFs that directly encode temporal changes into a learned latent vector, this deformation-based formulation provides a more structured way to model motion, effectively decoupling temporal variations from spatial variations. As a result, it reduces the risk of overfitting to per-frame geometry and appearance, often leading to sharper reconstructions and more photorealistic dynamics.

In practice, many dynamic scene reconstruction methods rely on calibrated and synchronized multi-camera capture systems operating in controlled environments, which limits their scalability and accessibility. To facilitate broader adoption, recent research has focused on techniques that are robust to a sparse number of views and tolerant of noisy camera parameters. This more challenging setting introduces stronger ill-posedness to the problem, requiring not only more careful regularization techniques, but also algorithmic breakthroughs for the representation of dynamic scenes. A seminal contribution in this direction is Nerfies [40], which addresses the problem of dynamic scene reconstruction from a single handheld mobile device. Nerfies extends the previous formulation as follows:

$$D_\Psi : (\mathbf{x}, \mathbf{z}_t) \mapsto \delta\mathbf{x} \qquad F_\Theta : (\mathbf{x} + \delta\mathbf{x}, \mathbf{d}, \mathbf{w}_t) \mapsto (\mathbf{c}, \sigma)\,, \tag{4.4}$$

where $\mathbf{z}_t$ is a per-frame learned latent code modulating the deformation field to capture time-dependent motion, and $\mathbf{w}_t$ is a learned appearance code accounting for per-frame radiance variations due to motion-induced appearance changes. To ensure physically plausible dynamics, Nerfies introduces an elastic regularization term that encourages local deformations to be as rigid as possible. Let $\mathbf{J} \in \mathbb{R}^{3\times3}$ be the Jacobian of the deformed coordinates (in the canonical space) with respect to the original coordinates (in the observation space), which can be computed via automatic differentiation. Given the singular value decomposition of the Jacobian $\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathrm{T}$, the proposed framework imposes rigidity of a transformation by penalizing its Jacobian's deviation from the identity matrix $\mathbf{I} \in \mathbb{R}^{3\times3}$ as

$$L_{\text{elastic}}(\mathbf{x}) = \| \log \mathbf{\Sigma} - \log \mathbf{I} \|_\mathrm{F}^2 = \| \log \mathbf{\Sigma} \|_\mathrm{F}^2\,. \tag{4.5}$$

In this chapter, we leverage multiview video data, which makes our reconstruction problem significantly less ill-posed compared to monocular approaches like Nerfies [40]. As a result, we follow the dynamic NeRF formulation in Equation 4.2, where we learn per-frame latent codes that condition the dynamic portion of our model.

4.2.2 Generative Radiance Fields

Most NeRF-based approaches in early 2020s focus on scene-specific optimization, where a single neural network is trained to represent a single static scene using dense multi-view supervision [135]. While these methods demonstrate impressive results, their limited generalization prompted a shift toward *generative* radiance fields that can synthesize novel 3D content.

Inspired by the success of generative models in 2D image synthesis, several works have explored 3D scene generation by training models on large collections of 2D observations of different classes of scenes, such as human faces. One of the earliest approaches in this area is GRAF [237], which introduces a conditional NeRF framework trained adversarially, where each scene representation is conditioned on latent shape and appearance codes. Building on this idea, GIRAFFE [311] handles more complex scenes consisting of multiple objects by introducing a compositional architecture. To further enhance the model expressivity, pi-GAN [238] introduces a model based on sinusoidal activations [312] and per-layer conditioning mechanism via Feature-wise Linear Modulation (FiLM) [313]. These NeRF-based generative models significantly outperform earlier 3D-aware frameworks based on discrete volumetric representations, such as HoloGAN [230], which use low-resolution voxel grids. The transition from voxel grids to continuous fields represents a pivotal shift in 3D generative modeling, giving rise to a new paradigm where high-quality, view-consistent 3D scenes can be synthesized from purely 2D supervision.

Training large-scale generative models of complex signals, such as 2D images or 3D scenes, requires synthesizing a vast number of samples during training. In the case of NeRFs, rendering a single image entails sampling and integrating many points along each camera ray, leading to prohibitively high computational costs. This bottleneck severely limits the scalability of generative NeRF-based methods, impeding both training efficiency and the quality attainable by such models. To address this challenge, a number of architectural changes have been proposed to make NeRF-based representations more computationally tractable. Among these approaches, StyleNeRF [11] draws inspiration from style-based 2D image generators and renders low-resolution feature maps that are subsequently upsampled and decoded into RGB images, enabling faster and more memory-efficient synthesis. EG3D [10] introduces a tri-plane representation, which factorizes the 3D volume into three orthogonal 2D feature planes, allowing for real-time rendering of novel 3D scenes. Complementing these advances, GRAM [307] decomposes the 3D scene into a collection of 2D implicit surfaces. These *radiance manifolds* dramatically reduce the number of samples required per ray while preserving high-fidelity volumetric effects. As radiance manifolds serve as the foundational representation in our efficient dynamic face rendering pipeline, we dedicate a separate subsection to detailing their formulation and implementation.

4.2.3 Generative Radiance Manifolds

The original NeRF formulation [49] defines radiance fields over a continuous 3D volume, where rendering involves sampling many points along each camera ray. These samples can lie anywhere within the volume, resulting in high computational cost due to dense volumetric integration. To improve the rendering efficiency, radiance manifolds [307] concentrate sample points onto a sparse set of 2D surfaces embedded within the 3D space. Here, each surface is implicitly defined as a level set of a scalar field represented with a manifold predictor $\mathcal{M}_\Phi$, which is parameterized by an MLP with weights $\mathbf{\Phi}$ as

$$\mathcal{M}_\Phi : \mathbb{R}^3 \to \mathbb{R}\,, \tag{4.6}$$

which takes in a point $\mathbf{x} \in \mathbb{R}^3$ and outputs a scalar value $s \in \mathbb{R}$. To define N radiance manifolds within the volume, N target values $\{s_i\}_{i=1}^N$ are selected. The i-th surface $\mathcal{S}_i$ is then implicitly defined as a zero-level set

$$\mathcal{S}_i = \{\mathbf{x} | \mathcal{M}_\Phi(\mathbf{x}) = s_i\}. \tag{4.7}$$

For rendering, given a camera ray with origin $\mathbf{o} \in \mathbb{R}^3$ and direction $\mathbf{d} \in \mathbb{R}^3$, we seek the intersection point $\mathbf{x}_i$ of the ray with surface $\mathcal{S}_i$. Assuming dense sampling along the ray, we identify the two closest samples $\mathbf{x}_f$ and $\mathbf{x}_b$ located at the front and the back of the surface, such that:

$$\mathcal{M}_\Phi(\mathbf{x_b}) = s_b \le \mathcal{M}_\Phi(\mathbf{x_i}) = s_i \le \mathcal{M}_\Phi(\mathbf{x_f}) = s_f\,. \tag{4.8}$$

Following the differentiable rendering framework of Niemeyer et al. [314], the intersection $\mathbf{x}_i$ is approximated using a single iteration of the secant method:

$$\mathbf{x}_i = \frac{s_i - s_b}{s_f - s_b}\mathbf{x}_f + \frac{s_f - s_i}{s_f - s_b}\mathbf{x}_b. \tag{4.9}$$

This approximation can be refined through additional iterations, though at an increased computational cost. Once the intersection points $\{\mathbf{x}_i\}_{i=1}^N$ are computed across the manifold set, radiance and density values are queried from a *conditional* radiance field:

$$F_\Theta : (\mathbf{x}, \mathbf{d}, \mathbf{z}) \mapsto (\mathbf{c}, \sigma) \tag{4.10}$$

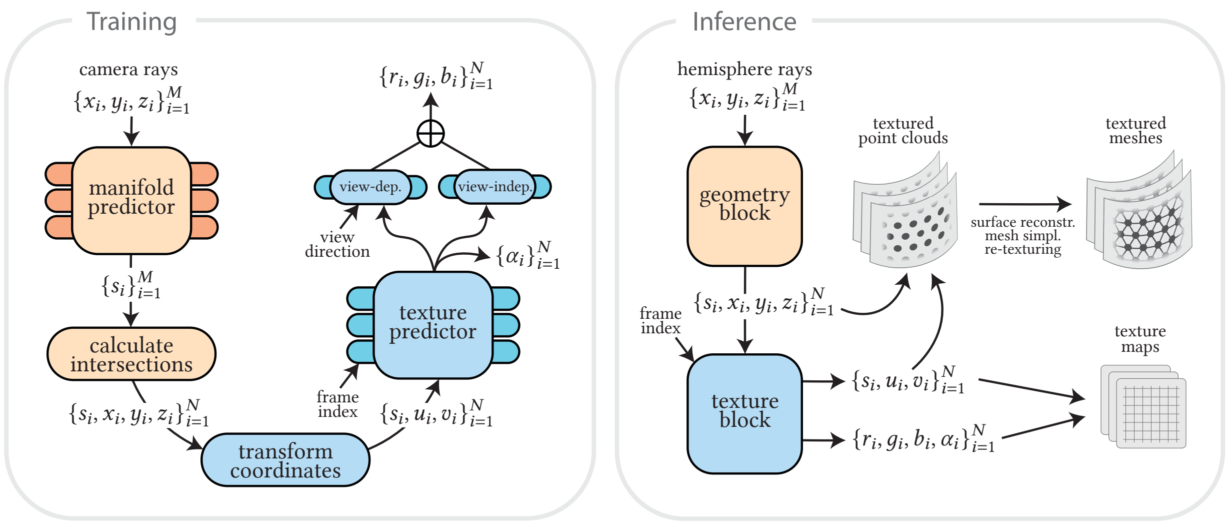

Figure 4.2: **Training and inference pipelines.** Given a set of rays from the training cameras, we determine the intersection of these rays with a set of implicit manifolds predicted by a single manifold predictor. After transforming these intersections to UV-space coordinates, a texture predictor outputs RGBA texture maps conditioned on the video frame index. At inference time, we shoot rays from the surface of a designated hemisphere around the scene towards its center, obtaining a single geometry and a video texture. The view-dependent branch is bypassed to ensure that the appearance is fully diffuse.

where $\mathbf{z} \in \mathbb{R}^{|\mathbf{z}|}$ is a latent code representing scene-specific attributes. Although point samples lie strictly on the learned surfaces, the radiance field F_Θ remains defined over the full 3D domain, ensuring continuity and compatibility with NeRF's volumetric rendering formulation. Final color values are obtained by integrating radiance along rays using the standard alpha-compositing procedure employed in NeRF, as described in Equation 2.18.

4.3 Methodology

In this section, we first formulate the objective problem and present an overview of our pipeline. After describing how we process our datasets, we elaborate on how we leverage radiance manifolds to learn efficient 3D representations from multi-view videos. Finally, we describe how we export our representation to a single set of textured meshes that can be rendered natively on traditional graphics software while maintaining the rendering quality.

4.3.1 Definitions and Overview

Our objective is to learn a volumetric 3D representation of a subject that can be played back on game engines without any special neural network integration. Given a multi-view video of a subject with K frames, we learn a static geometry and a dynamic appearance model in an end-to-end fashion. We take inspiration from recent advances in implicit geometry representations [113, 115], which significantly outperform explicit 3D reconstruction techniques that rely on mesh or point cloud representations. In our pipeline, similar to GRAM [307], the geometry is modeled by a set of 2D manifolds, embodied as a set of implicit surfaces. But unlike GRAM, the appearance is learned as a UV-mapped dynamic radiance over these manifolds, instead of the 3D xyz-space. We learn N distinct manifolds defined implicitly by a single manifold predictor

$$\mathcal{G} : \mathbb{R}^3 \to \mathbb{R}\,, \tag{4.11}$$

which takes in a point $(x, y, z) \in \mathbb{R}^3$ and outputs a single scalar $s \in \mathbb{R}$. Given a set of fixed scalars $\{s_i \in \mathbb{R} \mid i \in \mathcal{I} \triangleq \{1, 2, \ldots, N\}\}$, which we refer to as *s-values*, the manifold predictor defines a set of N isosurfaces that represent our static geometry:

$$\mathcal{S}_i = \{(x, y, z) \mid \mathcal{G}(x, y, z) = s_i\}\,. \tag{4.12}$$

In our appearance model, we first transform points on each manifold to UV-space coordinates via a fixed function $f : \mathcal{S}_i \to [-1, 1]^2$. For each manifold $i \in \mathcal{I}$ and each frame $j \in \mathcal{J} \triangleq \{1, 2, \ldots, K\}$, a texture predictor $\mathcal{T}_{ij}$ defines RGB and transparency fields as

$$\mathcal{T}_{ij} : [-1, 1]^2 \to \mathbb{R}^4,\;\; i \in \mathcal{I}\;\&\; j \in \mathcal{J}\,, \tag{4.13}$$

which takes a single UV-coordinate $(u, v) \in [-1, 1]^2$ outputs $(r, g, b, \alpha) \in \mathbb{R}^4$. Note that we do not estimate volume density as is the case for traditional volume rendering, but instead model 3D point transparency with an alpha value. This makes the radiance accumulation independent of the ray path, which is crucial for enabling the next step of exporting the learned manifolds as textured mesh layers. Our approach can be treated as a generalization of the multi-plane image representation [315], where we optimize arbitrary 2D surfaces instead of planes.

Once the training is completed, we collect samples across each manifold at a specific resolution and export these collections of 3D points, UV-coordinates, and RGBA values as a single set of topologized triangle meshes with UV-textures that can be efficiently rendered on legacy graphics renderers. We illustrate our training and inference pipelines in Figure 4.2.

4.3.2 Dataset

We use the publicly available dataset Multiface [316], where we follow a tailored preprocessing pipeline to ensure its compatibility with our model architecture and training. In particular, we gather multi-view videos of 3 subjects from V1 of the dataset (subject IDs V1 002914589, 002643814, 5372021), and 2 subjects from V2 (subject IDs 002421669, 002645310), ensuring that the subjects exhibit diverse facial geometries and appearances. To demonstrate sufficiently long facial performances, we pick $K = 60$ consecutive frames from each video sequence where subjects perform expressions freely. For all subjects, we scale and transform the scene parameters such that the subjects are centered at the origin and oriented along the positive x-axis with up-vector aligned with the positive z-axis, and that the camera centers are distributed roughly 1 unit away from the subjects. We discard the cameras with elevation angles of more than 45° and azimuth angles of more than 90°, which yields a set of cameras in the $x > 0$ half-space. For each subject, we also hold out 2 cameras to perform quantitative evaluations. This yields us 23 training cameras for V1 subjects and 50 training cameras for V2. Finally, we downsample all images to 768×500 resolution while adjusting the camera parameters accordingly, which provides an effective compromise between our training efficiency and ability to showcase high-resolution renders. We do not perform any background masking as our method is able to separate the foreground significantly either by restricting the scene volume or by placing it into the view-dependent component of radiance, which is discarded at inference time. We elaborate on this phenomenon in the next subsection.

4.3.3 Model Architecture and Training

Given K frames from a multi-view video with camera parameters, we sample M points uniformly along each camera ray and compute the intersections between these rays and the manifolds using the differentiable ray-manifold intersection algorithm [314] adopted in GRAM [307]. We sample $M = 256$ points along each ray, set the number of manifolds to $N = 12$, and train our model using videos consisting of $K = 60$ frames for each subject.

Previous techniques that model dynamic scenes with radiance manifolds [317, 318] have used explicit learned deformation of the manifolds to model scene animation. On the contrary, we model all frames of a dynamic sequence with a single set of static manifolds, which poses a non-trivial challenge. To achieve this, our technique uses a unique sequence of steps, where we 1) transform intersection points to UV-space, 2) separate RGB predictions into view-independent and view-dependent components, and 3) estimate the transparency of each intersection directly without computing volume densities. Given an intersection $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and a fixed center $\mathbf{c} \in \mathbb{R}^3$ of a unit sphere, we first project the intersection to the surface of the sphere and obtain $\mathbf{p}' \triangleq (x', y', z') = (\mathbf{p} - \mathbf{c})/\|\mathbf{p} - \mathbf{c}\|$. We then calculate the UV-space coordinates as $u = \frac{2}{\pi}\sin^{-1}(z')$ and $v = \frac{2}{\pi}\tan^{-1}(y'/x')$. The texture predictor receives UV-coordinates and the s-values of the intersections, as well as the frame index that is mapped to a learned latent code that conditions the predictions. The texture predictor is then branched into two layers that predict single-channel view-dependent compontent and three-channel view-independent component, former of which is conditioned on the view direction. The outputs of these branches are added together to produce the final RGB prediction. Such architecture allows us to discount view directional effects at inference time and achieve view-consistent rendering of exported meshes. Furthermore, it also helps us to separate most of the background from foreground by attributing the background to view-dependent component, particularly if the background is primarily grayscale, thus eliminating the need for explicit background removal. Finally, the alpha values are predicted as the raw output of our texture predictor, which can be directly used to alpha-composite our N-layered representation.

4.3.3.1 Training details

To promote training stability, we adopt the manifold initialization technique used in GRAM [319] and begin training with sphere-like manifolds centered at $\mathbf{c}$. We optimize our model in an end-to-end fashion by adopting ℓ_1 loss between the predicted and ground truth pixel values. To ensure that the appearance is mostly explained by the view-independent component, we penalize the output of the view-dependent branch with ℓ_2 penalty. To promote more stability, we apply ℓ_2 regularization to all manifold predictor layers except for the final one. The manifold and texture predictors are optimized jointly by minimizing the loss function

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{vd}}\mathcal{L}_{\text{vd}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}}\,, \tag{4.14}$$

where $\mathcal{L}_{\text{rec}}$ is the ℓ_1 reconstruction loss, $\mathcal{L}_{\text{vd}}$ is the view-dependent branch penalty, and $\mathcal{L}_{\text{reg}}$ is the manifold regularization with $\lambda_{\text{vd}} = 1.0$ and $\lambda_{\text{reg}} = 0.0001$. The manifold and texture predictors are jointly optimized using the Adam optimizer [262] with initial learning rates of 0.0007 and 0.0010, and exponential decay rates of 0.05 and 0.20 per 200 000 iterations, respectively. Using a batch size of 32 768 rays sampled across all training frames and views, we perform training for 500 000 iterations for each subject.

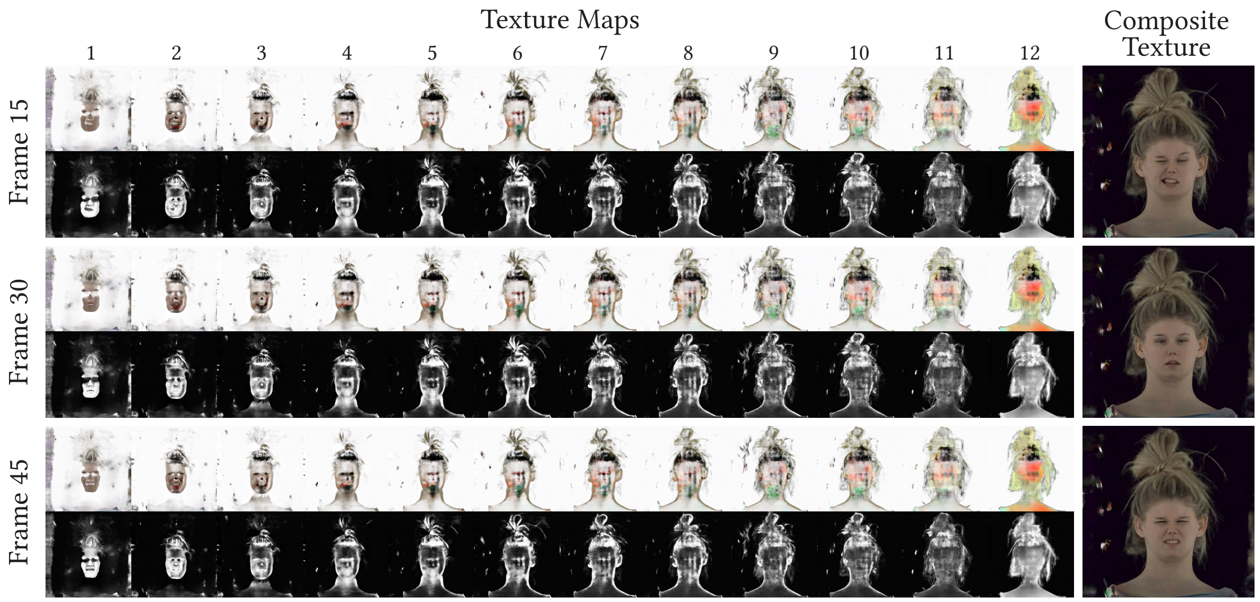

Figure 4.3: **Video texture visualization**. We illustrate 3 frames (15th, 30th, and 45th frames) from the learned texture video of subject 002914589. For each frame, we show full RGBA and alpha-only UV-space texture maps in the top and bottom rows, respectively.

4.3.3.2 Architecture Details

The manifold predictor is implemented as an MLP with 3 hidden layers of widths 128 and a final layer, where we choose the set of fixed scalars $\{s_i\}_{i=1}^{N}$ so that the initial concentric and sphere-like surfaces roughly falls within ±0.03 units of the surface of the face. These scalars are tuned slightly for each subject according to the size of the faces inferred by the tracked meshes provided in the Multiface dataset [316]. The texture predictor is implemented as an MLP with 8 hidden layers of widths 256 and 2 final layers corresponding to view-independent and view-dependent branches. Both input points and view directions undergo positional encodings and the frame indices are mapped to 32-dimensional learned latent codes through an embedding layer. Encoded input points and frame indices are fed into the MLP at its first layer whereas the encoded view directions are concatenated to the input to the view-dependent branch.

4.3.4 Exporting Layered Meshes and Textures

At test time, we gather points across the unit *hemisphere* by collecting azimuth and elevation angles in $[-\pi/2, \pi/2]$ uniformly at resolution $R \times R$ and shoot rays towards the sphere center $\mathbf{c}$. We set this center 0.25 units away from the scene center in the direction of negative x-axis to ensure that the entire scene is encompassed by the hemisphere. This gives us $R \times R$ samples across each manifold with UV-coordinates that are distributed uniformly in $[-1, 1] \times [-1, 1]$. For each frame, these points are used to query the texture predictor to yield N RGBA texture maps at resolution $R \times R$, where the view-dependent branch is bypassed to ensure the texture maps are fully diffuse. Our simple spherical projection yields reasonable texture mapping despite slight distortions near the edges of the maps [320]. We emphasize that we export the alpha channel as 8-bit images and hence store them efficiently without sacrificing the visual quality. Finally, while

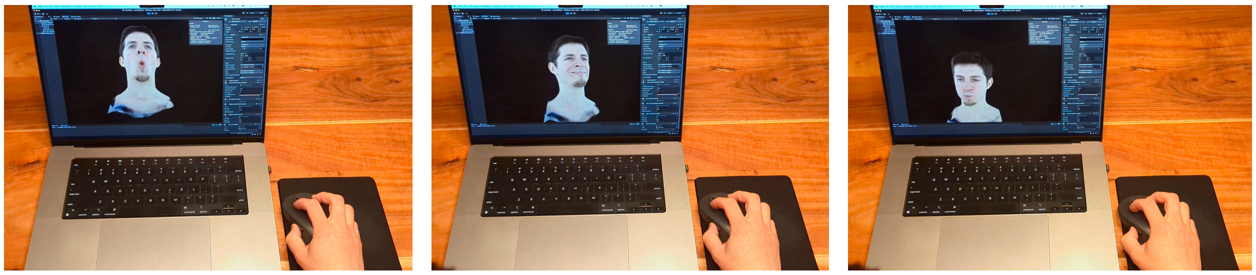

Figure 4.4: **Free-viewpoint rendering on Unity.** Our representation allows for free-viewpoint rendering of dynamic 3D volumes on consumer hardware.

the texture maps vary according to their frame indices, the geometry is the same across all frames.

To export our manifold-based representation to explicit surfaces, we reconstruct meshes from each of the N point clouds of size $R \times R$ via Poisson surface reconstruction [103], where the normals for each point are computed with respect to their neighboring points. We then simplify these meshes using a mesh decimation algorithm to reduce the number of vertices to a target mesh resolution $R^{\text{m}} \times R^{\text{m}}$. Finally, for each vertex in the simplified mesh, we determine the nearest point in the original point cloud and assign its corresponding UV-coordinate. The texture maps, on the other hand, can be downsampled to a specific target resolution $R^{\text{t}} \times R^{\text{t}}$ to meet the memory requirements of the renderer. To summarize, our final assets are: 1) a single set of N triangle meshes, each with number of vertices less than the target resolution $R^{\text{m}} \times R^{\text{m}}$ and 2) K sets of N RGBA texture maps at resolution $R^{\text{t}} \times R^{\text{t}}$ that form a UV texture video. We illustrate 3 frames from an example texture video along with composited texture maps in Figure 4.3.

4.3.5 Rendering on Game Engines

Our rendering pipeline in Unity using the exported layered mesh and texture sequences runs in real-time. We leverage two-pass deferred shading [321] on the GPU. When given a camera pose at runtime, we generate N G-buffers by shading each mesh layer and its opacity in a single render pass. Modern game engines use multiple render targets (MRT) for this purpose and we used culling masks to achieve this in Unity. For a small number of layers (*e.g.*, $N < 16$, which is the maximum texture sampler count supported in Unity), we composite G-buffers on the GPU by tracing a ray through all layers in one pass, similar to accumulating luminance in the traditional volume rendering pipeline. For more than 16 layers, we suggest using a prefix sum algorithm [322] on the GPU for efficient layer compositing.

In our experiments with $N = 12$, we achieved real-time performance on a 2019 Macbook Pro with an M1 Max chip and Unity 2021.3. This was consistent across five datasets and over 1,000 frames. The average rendering time per frame was under 17 ms (above 60 FPS) at a rendering resolution of 2560×1440, even for our largest reconstructed mesh of 6.3M triangles.

4.4 Experiments and Results

We evaluate our method on several subjects from the Multiface dataset [316], where we provide qualitative and quantitative comparisons against state-of-the-art neural rendering methods. We

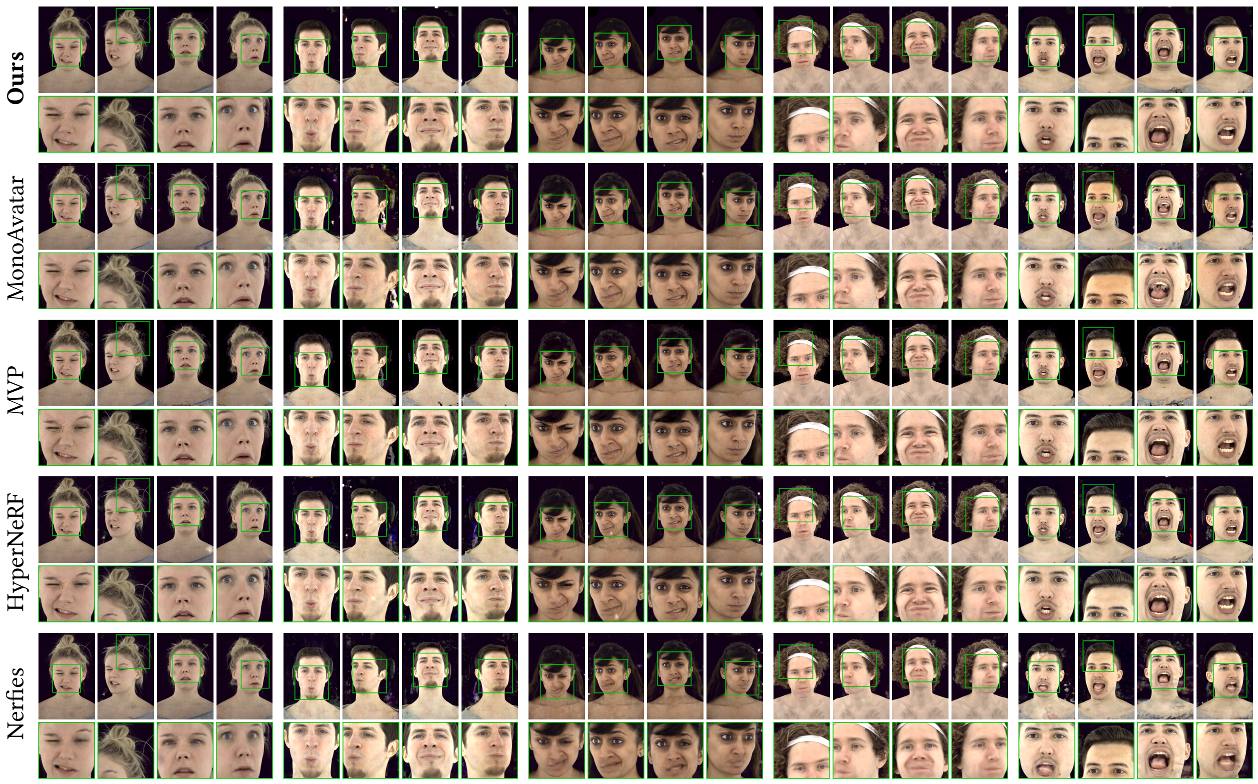

Figure 4.5: **Qualitative comparisons.** Our method achieves comparable visual quality to state-of-the-art neural rendering techniques while facilitating very efficient rendering of dynamic sequences on legacy graphics software without any custom integration of ML pipelines.

then perform more analysis of the configurability of our representation by assessing its performance with respect to varying numbers of manifolds, mesh resolution, and texture resolution.

4.4.1 Qualitative Results

We train our pipeline individually on 5 multi-view video sequences from [316], and illustrate our novel view synthesis results in Figure 4.5. Here, we provide comparisons against 4 state-of-the-art neural rendering methods—MonoAvatar [139], MVP [301], HyperNeRF [41], and Nerfies [40]. Despite discretizing the 3D volume into only $N = 12$ manifolds and hence sampling much fewer points across the scene during both training and evaluation, our approach manifests a comparable performance against other techniques. Furthermore, our technique does not require any MLP query or a custom pipeline during rendering and thus can be exported into a game engine, where we can perform free-viewpoint rendering of a dynamic 3D scene. We import our layered meshes and UV-textures into Unity and achieve the results demonstrated in Figure 4.4.

By interpolating between the learned latent codes of different frames at inference time, we can render our representation at higher frame rates to enhance the overall visual quality. We depict our frame interpolation results and provide comparisons in Figure 4.6, where we observe comparable performance against other methods.

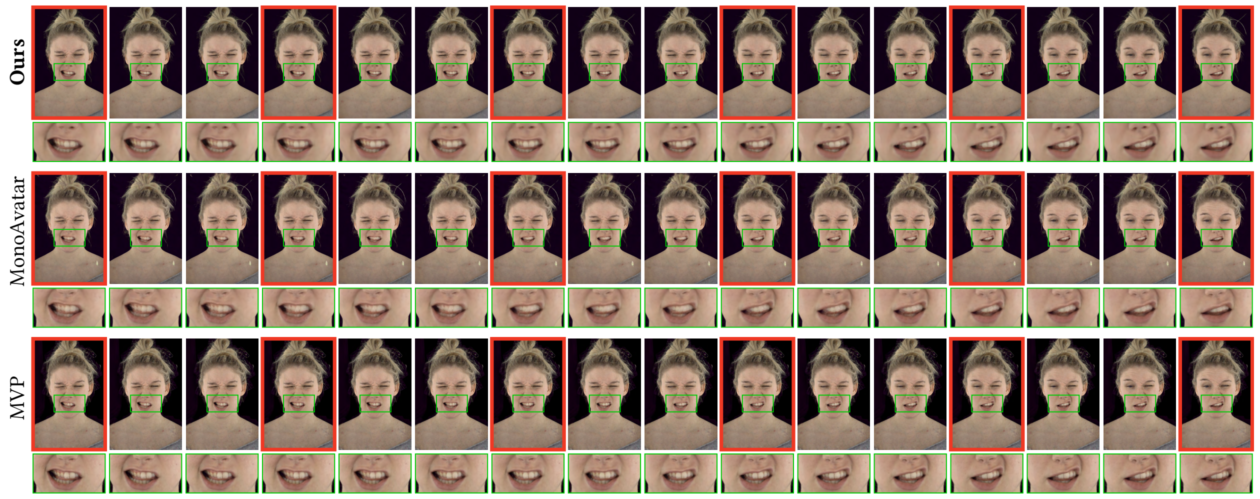

Figure 4.6: **Frame interpolation results.** Interpolating between the learned latent codes of frame indices allows us to achieve high-quality temporal interpolation between training frames with comparable performance to other approaches. Original frames are highligted in red.

4.4.2 Quantitative Results

For each subject, we perform quantitative evaluations on 2 held-out cameras across all $K = 60$ frames, totaling 120 images. In Table 5.1, we report average image quality metrics for our method and other methods in PSNR, SSIM [273], and LPIPS [274], where we consistently observe comparable performance across all methods. We also report VRAM usage, required disk storage, and frame rates for each of the methods, where we compress individual texture maps into a video and apply mesh compression to individual meshes using Draco[1] with no quantization of vertex positions and texture coordinates, and using the lowest compression amount. From our results, we observe that our method is able to achieve higher frame rates despite utilizing a comparable amount of storage against other methods. Note here that other methods can be run with much lower VRAM usage by reducing the batch size down to single ray per batch.

All ML training, including ours and the state-of-the-art methods, was done on a workstation with NVIDIA V100 GPU. Since the state-of-the-art methods require a Linux workstation with NVIDIA GPU also for inference, they were evaluated and profiled on this same workstation. Our method does not require such special ML integration, and we perform our evaluation on the Unity game engine on a 2019 Macbook Pro laptop.

4.4.3 Ablation Studies

Since the radiance manifolds constrain 3D volumes to a number of implicit surfaces, the rendering quality is strongly influenced by the number of manifolds chosen before training. We provide qualitative and quantitative comparisons across different numbers of manifolds in Figure 5.11 and Table 4.2. Here, we observe that not only the rendering quality suffers with decreasing number of manifolds, but also, capturing volumetric effects requires a sufficient number of samples.

Our exported representation allows for trading image quality with memory efficiency by

[1]https://google.github.io/draco/

Table 4.1: **Quantitative comparisons.** Our method attains comparable visual quality across various metrics while utilizing significantly less VRAM and enabling much higher frame rates. The image quality metrics are averaged over a total of 600 test images of all 5 subjects.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	VRAM ↓	Disk ↓	FPS ↑
Ours	25.49 ± 3.16	0.788 ± 0.069	0.356 ± 0.038	602 MiB	118 MiB	> 60
MonoAvatar [139]	24.59 ± 1.71	0.786 ± 0.042	0.341 ± 0.018	2746 MiB	296 MiB	0.33
MVP [301]	26.20 ± 2.34	0.738 ± 0.54	0.313 ± 0.025	1412 MiB	356 MiB	12.7
HyperNeRF [41]	26.91 ± 3.14	0.845 ± 0.0394	0.305 ± 0.041	2861 MiB	15 MiB	0.02
Nerfies [40]	26.11 ± 3.15	0.815 ± 0.042	0.332 ± 0.044	4205 MiB	15 MiB	0.03

Table 4.2: **Ablation on number of manifolds.** While significant gains in memory efficiency can be achieved by reducing the number of manifolds, it has a noteable effect on the visual quality. Numbers are averaged over a total of 240 test images of two subjects with IDs 002914589 and 002421669. We report total disk storage required by the meshes and the video texture and the total number of triangles in all meshes.

Num. Manifolds	PSNR ↑	SSIM ↑	LPIPS ↓	Num. Tri. ↓	Disk ↓
12	25.54	0.760	0.341	6264412	125.1 MB
4	24.47	0.728	0.372	2088497	42.9 MB
1	22.51	0.704	0.395	522242	11.0 MB

performing standard operations such as mesh simplification and texture downsampling. After reconstructing a single set of meshes via Poisson surface reconstruction [103], we decimate each of these meshes to meet a target number of vertices. We illustrate qualitative and quantitative evaluations for varying mesh resolutions in Figure 5.11 and Table 4.3. We observe that the image quality does not undergo a significant drop until 16×16 resolution per surface. This is because our layered mesh representation does not manifest high-frequency changes while still allowing for state-of-the-art rendering quality via learned alpha-manifolds. This provides us with an extremely lightweight geometry representation without sacrificing any visual quality or volumetric effects.

The texture resolution, on the other hand, naturally plays a vital role in rendering quality. To compare, we individually subsample each of the texture maps across all manifolds and frames bilinearly, and render each frame at original training resolution 768×500. We illustrate our results in Figure 5.11 and Table 4.4 where we observe a notable reduction in quality at 128×128 resolution.

4.5 Discussion

In this section, as in Chapter 3, we summarize our key takeaways from the developed methodology and subsequently reflect on the limitations of our approach, discussing potential research directions that could address these shortcomings. A broader outlook on the future of dynamic and animatable 3D face representations is deferred to the next chapter, following the development of our controllable volumetric avatar pipeline.

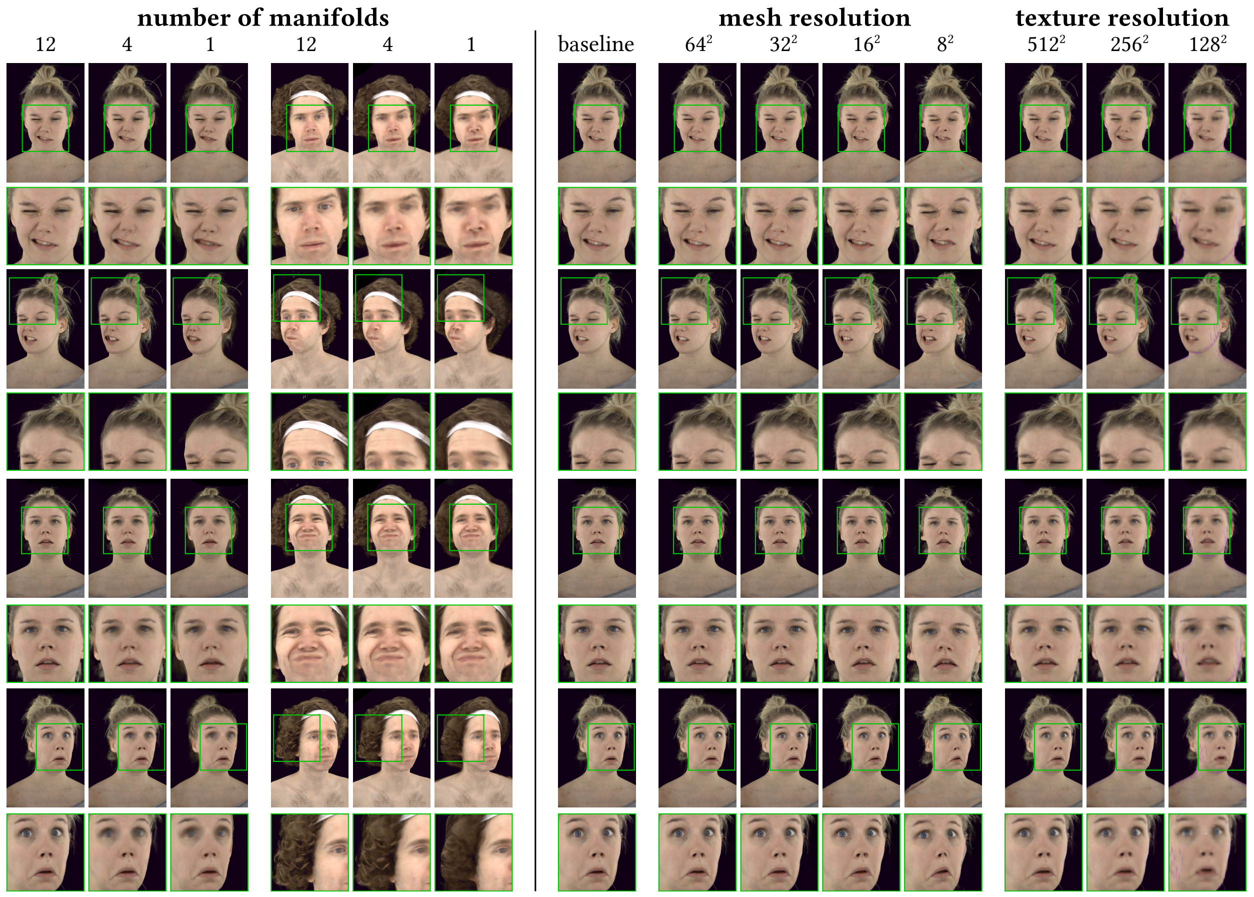

Figure 4.7: **Ablation results**. *Number of manifolds.* Using a sufficient number of manifolds is essential to attain photorealism and volumetric effects. *Mesh resolution.* We can decimate the exported meshes to much lower resolutions without sacrificing significant visual quality. *Texture resolution.* We can modify texture resolution arbitrarily at inference time to trade off image quality with rendering efficiency as desired.

In this chapter, we introduced a novel representation for high-quality, memory-efficient volumetric rendering of dynamic facial performances in legacy platforms. We demonstrated that a static layered mesh combined with dynamic RGBA textures offers several practical advantages without compromising visual fidelity, rendering speed, or memory usage. These assets are obtained by leveraging radiance manifolds [307] to model the dynamic performance as a set of layered, continuous 2D surfaces, which are subsequently discretized after training. Once generated, our representation requires no machine learning operations or complex computations at inference time, enabling efficient real-time rendering using standard graphics software on consumer-grade hardware. Our approach effectively compresses the full 3D dynamics into a sequence of 2D videos, allowing for trivial streamability using conventional video compression and streaming protocols.

One key question addressed in this chapter is whether volumetric effects, as well as complex geometric and appearance variations of a 3D face, can be faithfully simulated by offloading all dynamic components to the appearance model. Our findings suggest that this is indeed feasible: with a moderate number of layered surfaces, volumetric effects can be rendered at state-of-the-art quality, and intricate dynamics can be synthesized through alpha composition of these surfaces, eliminating the need to explicitly learn a volumetric density field, as required in traditional radiance field pipelines. Furthermore, we observe that discretizing continuous surface representations into meshes and converting texture fields into pixel-space texture maps at moderate resolutions preserves overall visual fidelity. Notably, the learned geometry of the radiance manifolds is designed to be smooth, allowing us to maintain high-quality reconstruction even under aggressive mesh simplification, further improving memory and rendering efficiency.

Limitations. Our representation is designed to discretize 3D geometry and dynamics into conventional textured triangle primitives, enabling practical deployment on traditional graphics pipelines. However, since the static mesh does not precisely conform to the true face geometry, sampling across a discrete set of manifolds (as opposed to the full 3D volume) can lead to shell artifacts under extreme viewing angles, as illustrated in Figure 5.13(b). Our layered mesh with transparency can be interpreted as a generalization of Multiplane Imaging (MPI) [315], where we instead learn a set of surfaces that follow a coarse face geometry and represent dynamic content. Due to this coarse approximation, there exists an inherent limitation in the range of viewing angles for which artifact-free rendering is possible [323]. Our empirical findings suggest that such artifacts can be alleviated by carefully initializing the manifolds based on the scale of the scene and maintaining sufficiently small spacing between consecutive layers, while still ensuring that the spacing remains large enough to preserve volumetric effects. Beyond these heuristics, sophisticated geometry regularization techniques [112] can be employed to further refine the predicted surfaces. Ultimately, however, our findings reveal a fundamental trade-off: increasing the range of artifact-free viewing angles necessitates higher mesh resolution and more complex geometry, which comes at the cost of increased computational and memory demands.

Our pipeline captures view-dependent effects using a view direction–conditioned function, which offers high representational capacity and enables more faithful reconstruction of the input sequence. However, since this conditioning is implemented via an MLP, it is discarded when we export our final assets. As a result, our results do not exhibit view-dependent effects, as illustrated in Figure 5.13(a). If this conditioning mechanism were instead decoupled from neural network inference, *e.g.*, by querying learned spherical harmonic functions [143] with view directions in an offline manner, our pipeline could be extended to estimate specular maps as well. This would, in turn, enable plausible specular relighting within traditional graphics pipelines.

It is worth noting that our method samples across the 3D volume by casting a single ray per pixel and querying MLPs at multiple points along each ray. This approach is known to be susceptible to aliasing artifacts [302] and results in slow training [141]. Future work could integrate recent neural rendering frameworks that incorporate anti-aliasing techniques and fast, grid-based representations [324] to improve both performance and visual quality. However, adapting radiance manifolds to operate within such acceleration pipelines remains a nontrivial challenge.

Finally, we emphasize that the primary objective of this chapter is to evaluate the feasibility of static layered mesh representations for dynamic face synthesis. Our results demonstrate that these representations are indeed capable of supporting high-quality 3D playback of dynamic face sequences. However, we have not yet provided any insights on making these representations controllable, a key requirement for virtual telepresence applications, which serves as the central motivation for the work presented in the following chapter.

Table 4.3: **Ablation on mesh resolution.** Despite reducing the memory footprint of the geometry, visual quality is maintained for resolutions as low as 32×32. For all mesh resolutions, the texture resolution is set to 1024×1024 and requires 5.9 MB storage after video compression. Numbers are averaged over 120 test images of a single subject with ID `002914589`.

Mesh resolution	PSNR ↑	SSIM ↑	LPIPS ↓	Num. Tri. ↓	Disk (mesh) ↓
512×512	30.10	0.858	0.284	6261920	118.1 MB
256×256	29.58	0.838	0.293	1377119	22.2 MB
128×128	29.57	0.839	0.290	250740	4.1 MB
64×64	29.49	0.837	0.289	43224	721 KB
32×32	29.12	0.831	0.291	9962	170 KB
16×16	27.26	0.795	0.306	2574	38 KB
8×8	23.81	0.705	0.364	728	11 KB

Table 4.4: **Ablation on texture resolution.** We can reduce the memory footprint of the video texture by simply subsampling each frame at inference time. For renders of resolution 768×500, a noteable drop in quality occurs at 256×256 texture resolution. For all texture resolutions, the mesh resolution is set to 512×512, hence the number of triangles and disk storage for meshes are constant and are 6261920 and 118.1 MB, respectively. Numbers are averaged over 120 test images of a single subject with ID `002914589`. Note that the texture video size will increase with the number of frames in the input video.

Texture resolution	PSNR ↑	SSIM ↑	LPIPS ↓	Disk (texture) ↓
1024×1024	30.10	0.858	0.284	5.9 MB
512×512	30.29	0.859	0.303	2.1 MB
256×256	29.51	0.836	0.341	667 KB
128×128	27.32	0.791	0.404	183 KB

5 Synthesis and Animation of Volumetric Face Avatars

5.1 Introduction

Building on our insights from the previous chapter, we now shift our focus to *controllable* 3D face representations, where our objective is to develop a representation that can be driven by *some* description of facial expressions, such as the 3DMM parameters. A suitable motivating application for this is virtual telepresence, in which tracked face parameters from one client are used to drive a corresponding avatar on another client's device, enabling real-time, bidirectional communication. These emerging technologies offer immersive alternatives to traditional video-based communication, further blurring the line between physical and synthesized realities. Beyond virtual telepresence, drivable avatars hold transformative potential across industries such as education. For example, some institutions are already piloting spatial 3D teaching assistants that interact with students in VR-based classrooms [325, 326].

In this chapter, our motivation closely aligns with that of the previous one. Specifically, we aim to develop representations that maintain backward compatibility with legacy rendering hardware and software, support efficient and trivial streaming, achieve photorealistic quality, and exhibit a favorable memory-compute tradeoff. In addition to the challenges discussed in the preceding chapter, this chapter introduces the crucial challenge of controllable synthesis. This involves conditioning the model on tracked expression parameters and ensuring that it can generalize to unseen expressions beyond the training data, while preserving realism and identity consistency. To establish the problem setting, we assume that we are given multiview videos of a single subject, as in the previous chapter. However, in this case, each multiview frame is additionally annotated with facial expression parameters, which are assumed to be obtained via an offline face tracking algorithm. These expression labels will serve as conditioning inputs to enable controllable synthesis in the subsequent framework we introduce later in this chapter.

5.1.1 Main Challenges

A static layered mesh paired with dynamic textures provides a strong foundation for designing backward-compatible and streamable avatars. But it is important to note that, within the previous pipeline, the texture predictor is the sole dynamic component of the representation, as it is responsible for capturing all variations in geometry and appearance. For controllability, a straightforward approach would be to condition the texture predictor on expression parameters. However, a key constraint in our formulation is that this conditioning mechanism must remain exportable at inference time, that is, it must be implemented without relying on any neural network layers during deployment. This requirement introduces a non-trivial challenge that complicates the design of conventional conditioning strategies.

Suppose there exists a conditional texture mapping representation capable of synthesizing novel textures at inference time without using any neural networks. The central question then becomes: what should be the *capacity* of this model, and more fundamentally, is it possible to capture complex geometric and appearance variations solely at the texture level? A key observation is that certain geometric changes, such as mouth opening, can be easily achieved by directly deforming the mesh, *i.e.*, by moving the vertices around. However, in our formulation, the mesh is kept entirely static, which necessitates offloading such deformations to the texture mapping and texture coordinate look-up operations. Accomplishing this without significantly increasing the model's capacity poses a substantial challenge.

Finally, assuming we can achieve a reasonably compact static mesh representation, how can we ensure that it generalizes to novel expressions, allowing the avatar to be driven by arbitrary facial parameters within the training distribution? One potential approach is to leverage the tracked mesh itself, as the underlying parametric face model used to generate the mesh offers a strong prior for synthesizing new expressions. Indeed, this paradigm is commonly adopted, where volumes parameterized as NeRFs [49] or Gaussian primitives [50] are anchored to tracked meshes to support expression-driven synthesis [139, 156, 327]. However, since our framework prohibits explicit mesh deformation at inference time, we must instead achieve generalization directly within the lower-dimensional space of expression parameters, which presents a more challenging learning problem. Addressing this requires careful construction of the training dataset, along with loss functions and regularization strategies to guide the model toward generalization.

5.1.2 Main Objective and Contributions

In this chapter, given observations of a dynamic 3D face, our goal is to learn 1) 3D geometry and 2) appearance *models* that can be controlled by facial expression parameters in real-time, yielding photorealistic images, while being compatible with legacy graphics platforms and existing streaming infrastructure. Our key insight is to discretize the three constituent scene components—geometry, appearance, and deformation—and convert them into classical primitives, such as meshes and textures. To this end, we represent the canonical geometry of the face as a *static* layered mesh extracted from a learnable radiance manifold inspired by previous work [307, 308, 328]. We couple this mesh with a controllable appearance model driven by a 3D morphable face model [3]. Uniquely, we model deformations not in 3D geometry space, but instead as dynamic warp and texture fields defined in the UV space of the layered mesh. This design choice allows us to retain the static mesh, thereby enabling the trivial streamability of our controllable representation.

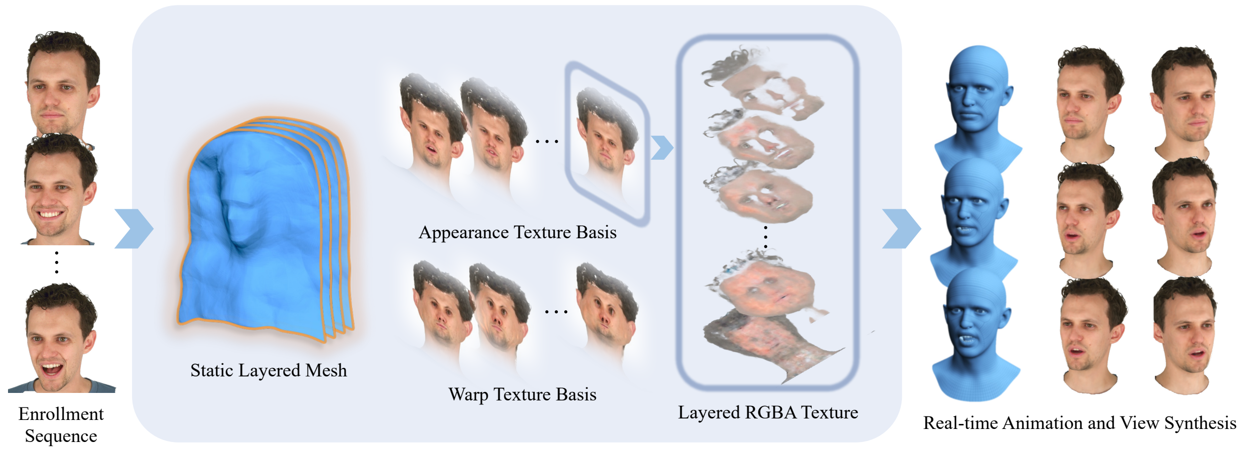

Figure 5.1: We introduce a novel representation for rendering animatable volumetric 3D face avatars using meshes and textures. We generate a layered mesh and blend-textures that model the appearance and deformations, and a simple linear transformation that maps tracked face model parameters to blend weights. Our representation can be deployed in traditional graphics pipelines efficiently and conveniently.

Moreover, by introducing a warp field, we can offload a significant amount of geometry variations to UV-space warps, substantially reducing the number and size of assets required at inference time. Please see Figure 5.1 for an overview of our pipeline.

To deploy our model without relying on custom rendering pipelines or neural networks, we represent warps and textures as combinations of blendable *fields*, where the blend weights are predicted via a simple linear transformation of expression parameters derived from a morphable face model. During enrollment, we jointly optimize radiance manifolds, a warp basis, a texture basis, and the associated linear mappings to reconstruct the captured enrollment sequence of a subject. At deployment time, the radiance manifolds are exported as a single static mesh, as in the previous chapter, while the learned warp and texture bases are discretized into sets of 2D warp and texture images in pixel space. Although it is not immediately obvious that discretizing continuous warp and texture fields at moderate resolutions would preserve visual quality, we empirically observe that such discretization introduces negligible artifacts. Using these assets, the animation and rendering can be performed with a simple warp-and-blend shader and standard rasterization on any graphics platform, as we demonstrate later in this chapter.

To help avoid overfitting to the expressions from the calibration sequence of the target subject, and thereby ensure generalization to novel expressions, we train each subject jointly with a set of synthetic subjects, whose expression parameters are sampled from a broader, more carefully constructed distribution. This is done in addition to standard regularization techniques such as weight penalties for the respective geometry, warping, and texture predictors.

5.1.3 Preliminaries

In this part of the thesis, we finally consider the original observation model in (2.25):

$$\mathbf{y} = \mathcal{R}(G(\mathbf{p}), A(\mathbf{p}); \mathbf{c}) + \mathbf{n}. \tag{5.1}$$

Here, we assume that we are given a set of multi-view observations of a single subject with different facial expressions, from known and calibrated cameras $\{\mathbf{y}_i, \mathbf{p}_i, \mathbf{c}_i\}_{i=1}^{N}$, where we estimate the facial expression parameters in each *frame* (with multiple views) using an off-the-shelf optimization algorithm that fits a 3DMM to our observations. We emphasize that not only we aim to learn geometry and appearance models that explain the observations under the given control inputs, but we also would like these models to generalize to novel inputs, which requires careful regularization of these respective models during training. In the remainder of this chapter, we use a notation consistent with the submitted work [329].

5.2 Background and Related Work

In this section, we review volumetric head avatar methods by categorizing them according to their underlying representations, including surface-based, NeRF-based, and point-based approaches.

5.2.1 Surface-based Avatars

Early real-time 3D avatar systems were predominantly built on parametric models, most notably 3DMMs [3, 9, 174]. Since these models are differentiable by design, they enable efficient inverse rendering pipelines that facilitate canonical expression capture and retargeting. However, their expressiveness is fundamentally constrained by their low-dimensional linear subspaces, which limit their ability to represent high-frequency geometric details, complex appearances, and view-dependent phenomena, ultimately compromising photorealism.

Recent advances in neural rendering have addressed these limitations by leveraging implicit surface representations, which model geometry as the zero-level set of learned continuous fields [113, 330]. Unlike mesh-based 3DMMs, these representations are not bound by a fixed topology and can accommodate challenging geometries in regions like hair or the oral cavity. Moreover, several works have demonstrated that implicit representations can be made controllable by using 3DMM parameters to condition the deformations of learned surfaces [331, 332], bridging classical model-based control with modern high-fidelity synthesis. Appearance modeling in these systems is similarly enhanced through learned radiance fields that enable view-consistent rendering.

To further enhance the rendering of volumetric effects, recent works have departed from the single-surface modeling and instead adopted multi-layered surface representations. Among these, BakedAvatar [318] leverages radiance manifolds [307] that are discretized into layered meshes, allowing efficient rasterization at inference time while preserving complex volumetric phenomena. Other methods augment mesh surfaces with volumetric primitives [333, 334], creating hybrid architectures that combine the benefits of explicit surface control with volumetric appearance modeling. As these approaches substantially deviate from purely surface-based representations, we discuss them in detail in subsequent sections.

5.2.2 Neural Radiance Field-based Avatars

On a fundamental level, animating volumetric avatars requires introducing a dynamic representation capable of generalizing to novel control signals at inference time. One formulation of such dynamic representations is exemplified by Nerfies [40], which models non-rigid deformations via learned latent codes conditioned through a deformation network and a canonical radiance field, as described in Equation 4.4. This approach can be extended to support controllable animation by directly conditioning the networks on a control input instead:

$$D_\Psi : (\mathbf{x}, \mathbf{p}) \mapsto \delta\mathbf{x} \qquad F_\Theta : (\mathbf{x} + \delta\mathbf{x}, \mathbf{d}, \mathbf{p}) \mapsto (\mathbf{c}, \sigma). \tag{5.2}$$

Here, the vector $\mathbf{p} \in \mathbb{R}^p$ represents control parameters such as 3DMM-based expression and pose coefficients. Similar conditioning mechanisms enable animation-driven synthesis and are adopted by several NeRF-based avatar systems in the literature [332, 335, 336].

While control inputs such as 3DMM coefficients provide a compact representation of dynamic mesh deformations, they are ultimately a low-dimensional proxy for complex high-resolution surfaces. To better capture fine-grained dynamics, an alternative class of methods anchors volumetric representations directly to tracked meshes in mesh space. Among these techniques, MonoAvatar [139] encodes per-vertex features on a tracked template mesh, which are used to interpolate color and density values from an MLP. Other approaches forgo MLPs entirely in favor of explicit volumetric grids to accelerate inference, albeit at the cost of increased memory consumption [301]. More recent works mitigate the memory-performance trade-off by leveraging the Instant-NGP framework [141], utilizing multi-resolution hash grids for compact and efficient neural field representation [327]. Another paradigm generalizes traditional blendshape models to volumetric settings. These methods synthesize dynamic effects by linearly combining a set of pre-learned volumetric *blend assets*. As with other dynamic avatar representations, these blend-based volumetric models have also been accelerated through hash-based data structures, yielding real-time performance without compromising quality [333].

5.2.3 Point-based Avatars

Suppose we aim to represent a controllable 3D head avatar *realization* via an oriented point cloud $\{\mathbf{x}_i, \mathbf{n}_i, \mathbf{f}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^3$ are positions, $\mathbf{n}_i \in \mathbb{R}^3$ are normals, and $\mathbf{f}_i \in \mathbb{R}^F$ are appearance-related features such as diffuse or specular albedo in physical color spaces, or more abstract representations in learned feature spaces that are subsequently mapped to RGB values. To control these attributes, we can define a geometry *model* $G(\mathbf{p})$ that transforms positions and normals in some canonical space $\{\mathbf{x}_i^c, \mathbf{n}_i^c\}_{i=1}^N$ to a deformed space $\{\mathbf{x}_i^d, \mathbf{n}_i^d\}_{i=1}^N$ given a control input $\mathbf{p} \in \mathbb{R}^p$. Such a transformation can be carried out by a learnable function parameterized as a neural network or the deformation function of a morphable face model can be used directly. Given a set of canonical features $\{\mathbf{f}_i^c\}_{i=1}^N$, an appearance *model* $A(\mathbf{p})$ may optionally transform these colors to another space $\{\mathbf{f}_i^d\}_{i=1}^N$ to represent deformation-induced appearance changes. Camera view-dependent effects can be carried out by the rendering function $\mathcal{R}$ that rasterizes points and composites colors to produce the final image.

Point-based representations with simple point primitives have been shown to effectively model volumetric effects while enjoying efficient rendering thanks to fast, GPU-accelerated rasterization [337, 338]. With the advent of 3D Gaussian Splatting (3DGS) [50], point-based avatars were extended to use anisotropic Gaussian primitives with per-primitive, view-dependent appearance attributes, leading to a higher representational capacity. These advancements have led to a wave of research into 3DGS-based avatars using animation models of varying characteristics.

Following the formulation in (2.23), consider a canonical Gaussian cloud $\mathcal{G} = \left\{\left(\boldsymbol{\mu}_i^c, \Sigma_i^c, \mathbf{f}_i^c, \alpha_i^c\right)\right\}_{i=1}^N$, where we replace the color attributes with more general appearance features $\mathbf{f}_i^c$. Under control parameters $\mathbf{p} \in \mathbb{R}^p$, a geometry model $G(\mathbf{p})$ can transform the positions, shapes, and opacities of the Gaussians to some deformed space, yielding $\{\boldsymbol{\mu}_i^d, \Sigma_i^d, \alpha_i^c\}$. As before, an appearance model $A(\mathbf{p})$ may optionally transform features to produce $\{\mathbf{f}_i^d\}_{i=1}^N$. Such a modeling paradigm underpins recent influential works such as GaussianAvatars [156] and Gaussian Head Avatar (GHA) [339]. Here, GaussianAvatars attaches Gaussian primitives to a FLAME template mesh, and rigs this mesh (along with the attached Gaussians) directly from tracked pose and expression coefficients, whereas Gaussian Head Avatars [339] uses a learned deformation function conditioned on tracked coefficients. Other approaches adopt alternative control mechanisms, such as Gaussian Blendshapes [334], which generalize mesh-based blendshape models to the Gaussian domain, enabling expressive animation by linearly blending a set of Gaussian basis components.

5.3 Methodology

A 3D avatar system generally consists of two phases, an *enrollment* phase in which the avatar is created using data captured from a subject, and a *deployment* phase in which the avatar is streamed, animated, and rendered on the client device from a desired viewpoint. Our goal is to design a volumetric avatar representation that is compatible with legacy rendering platforms during the deployment phase. We achieve this by exporting our enrolled volumetric avatar to classical graphics primitives like meshes and textures that can be rendered efficiently using simple programmable shaders on any graphics platform without additional custom engineering, agnostic of the underlying device hardware or software. We also aim to make our representation conducive to online streaming, which includes the ability to trade off quality and data bandwidth via data compression, similar to today's online video streaming systems.

5.3.1 Avatar as Layered Mesh

Given the calibration video of a subject, our objective is to learn a single layered mesh representation that can be dynamically textured using expression coefficients of a parametric face model in real-time. To achieve this, as done in Chapter 4, we model the geometry as a set of static 2D implicit surfaces. But this time, the appearance is modeled as a UV-mapped dynamic radiance controlled by 3DMM coefficients. To model the geometry, we learn N implicit surfaces $\{\mathcal{S}_i\}_{i=1}^N$ defined by a single manifold predictor $\mathcal{M} : \mathbb{R}^3 \to \mathbb{R}$, which takes in a 3D point $\mathbf{x} \in \mathbb{R}^3$ and outputs a scalar value $s \in \mathbb{R}$, as in the previous chapter. We map all points on these surfaces via a learnable function $f : \mathcal{S}_i \to [-1, 1]^2$ to obtain UV-coordinates, which allows the network to deviate from the deterministic spherical mapping used in Chapter 4. This is an important step to better learn dense UV-space correspondences across different expressions in a subject-specific way. In this UV-space, we learn a set of W warp fields and T texture fields for each surface by parameterizing the following functions as MLPs:

$$\mathcal{W}_{ij} : \mathbf{u} \mapsto \delta\mathbf{u} \tag{5.3}$$

$$\mathcal{T}_{ik} : \mathbf{u} \mapsto \mathbf{c}\,, \tag{5.4}$$

where $\mathbf{u} \in [-1, 1]^2$ is a UV-space coordinate, $\delta\mathbf{u} \in \mathbb{R}^2$ is a UV-space offset, $\mathbf{c} \in \mathbb{R}^c$ is a view-dependent color parameterized as spherical harmonics coefficients [143], and $i \in \{1, 2, \ldots, N\}$,

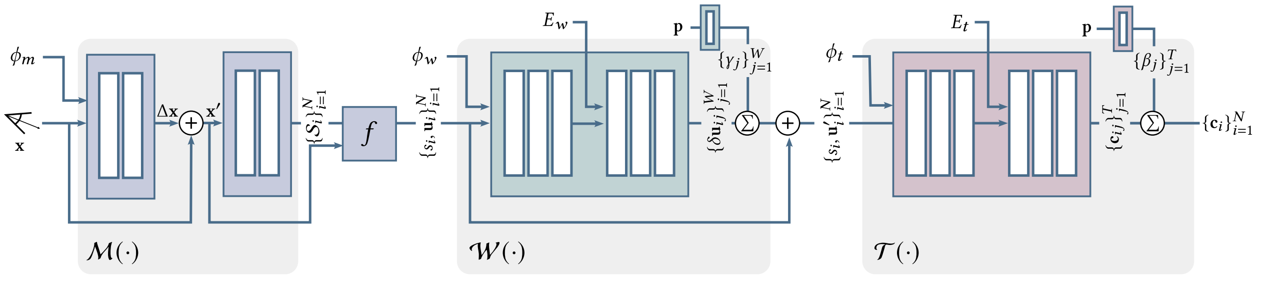

Figure 5.2: **Training pipeline for enrollment phase.** Our model consists of three separate modules: a manifold predictor $\mathcal{M}$, a warp predictor $\mathcal{W}$, and a texture predictor $\mathcal{T}$. Here, $\mathcal{M}$ is a scalar field that defines layered implicit surfaces. The intersections with these surfaces are spherically mapped to the UV-space via a learnable function f. Then, the output subsequently queries $\mathcal{W}$ to obtain a basis of UV-offsets. These offsets are then linearly blended as a function of expression parameters of a face model and added to the original values. Finally, the new coordinates are fed through $\mathcal{T}$ which predicts a basis of RGBA appearances which are also linearly blended as a function of expression parameters. Each module takes in learned latent codes ϕ_m, ϕ_w, ϕ_t for multi-subject training, while $\mathcal{W}$ and $\mathcal{T}$ takes in learnable embedding matrices E_w and E_t to output bases of warps and textures.

$j \in \{1, 2, \ldots, W\}$, $k \in \{1, 2, \ldots, T\}$. Given a set of face model parameters $\mathbf{p} \in \mathbb{R}^p$ of a single frame, a layered warp field $\{\mathcal{W}_i\}_{i=1}^N$ and a layered texture field $\{\mathcal{T}_i\}_{i=1}^N$ are obtained by

$$\mathcal{W}_i = \sum_{j=1}^{W} \gamma_j \mathcal{W}_{ij} \quad \text{and} \quad \mathcal{T}_i = \sum_{k=1}^{T} \beta_k \mathcal{T}_{ik} \tag{5.5}$$

where $\{\gamma_j\}_{j=1}^W$ and $\{\beta_k\}_{k=1}^T$ are a set of weights obtained as a learnable *linear* function of $\mathbf{p}$. Given a 3D point $\mathbf{x} \in \mathcal{S}_i$ and its UV-coordinate $\mathbf{u}$, we compute the warped UV-values $\mathbf{u}' = \mathbf{u} + \mathcal{W}_i(\mathbf{u})$ which are used to query the blended texture field to obtain color and transparency as $\mathbf{c} = \mathcal{T}_i(\mathbf{u}')$. We design our pipeline in a way that the implicit surfaces and the warp and texture bases can be efficiently discretized and exported into a layered mesh and UV-space maps in pixel-space so that our assets can be immediately deployed to any graphics platform without relying on custom rendering algorithms or incorporation of ML tools.

5.3.2 Dataset

To train our model, we use multiview videos from the NeRSemble dataset [340], which consists of a set of subjects with various facial expressions and talking sequences. To explicitly supervise correspondences between different frames, we fit a parametric face model to the video sequences of each subject to obtain per-frame expression coefficients as well as per-pixel UV-values Our 3DMM includes linear bases of identity and expression. For each frame, we fit the 3DMM by estimating 599 landmarks in 2D and optimizing identity, expression, rotation, and translation parameters of the 3DMM using a loss function that encourages consistency between per-vertex landmarks of the 3DMM and the observed 2D landmarks [341]. The parameter size of our expression model is $p = 63$. As a preprocessing step, we transform the camera extrinsics to align faces across frames to a canonical 3D face in order to account for strong head rotations during the capture. We

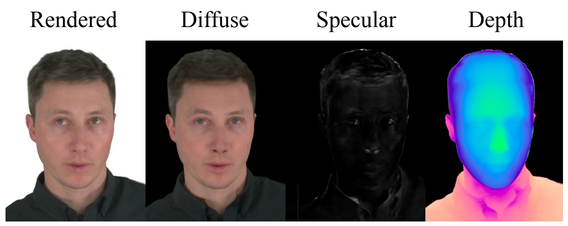

Figure 5.3: During training, the radiance is modeled using spherical harmonics coefficients, which can be decomposed into diffuse and specular components. The appearance can be exported as just diffuse or both diffuse and specular texture images.

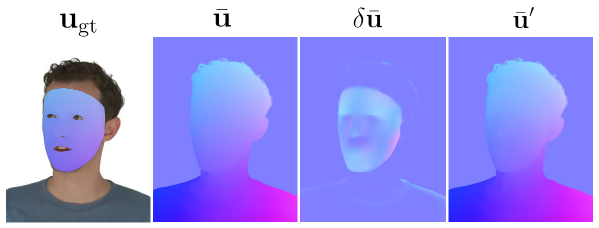

Figure 5.4: Given ground truth UV-coordinates $\mathbf{u}_{\text{gt}}$, our model is supervised to match the *expectation* of warped coordinates $\bar{\mathbf{u}}'$ to the ground truth. We visualize the expectations of the spherically mapped coordinates $\bar{\mathbf{u}}$ and the warps $\delta\bar{\mathbf{u}}$.

downsample the original images to a resolution of 802×550, which we found to offer an effective trade-off between training efficiency and the ability to demonstrate high-resolution visual effects.

A note on expression generalization. We again emphasize that learning an expressive geometry and appearance model that also generalizes to novel expressions outside of the calibration sequence is a challenging and ill-posed problem. Furthermore, sampling across 2D discrete surfaces instead of the entire 3D volume introduces more instability in training, which can cause shell artifacts in extreme poses, as seen in Chapter 4. To aid generalization to novel expressions and mitigate stability issues arising in the joint learning of geometry and appearance, we introduce a multi-subject training paradigm, where we use a publicly available synthetic multi-view multi-expression image dataset [342]. In our experiments, we combine each subject in the NeRSemble dataset [340] with synthetic subjects that have similar face shapes to the real subject, which helps avoid overfitting to the expressions from the calibration sequence of the target subject and prevents artifacts. We empirically found that training with 20 synthetic subjects strikes a good balance between training times and overall expression generalization.

5.3.3 Model Architecture and Training

Our architecture consists of three modules: a manifold predictor $\mathcal{M}$, a UV-space warp predictor $\mathcal{W}$, and a texture predictor $\mathcal{T}$, which we illustrate in Figure 5.2. While we note the similarity of this architecture to the one we develop in Chapter 4, there are several notable changes.

Manifold predictor. We implement our manifold predictor as a *subject-specific* scalar field that takes in 3D coordinates in xyz-space and a learned latent code ϕ_m for each subject. This module defines a set of N implicit surfaces, which are *static* for the entire sequence of a given subject. To achieve this, an input 3D point is first deformed using a subject-specific deformation field, and subsequently used to predict a scalar value that determines the manifold geometry. Given a camera ray, we use the ray-mesh intersection algorithm [314] to find the set of xyz-space intersections per ray and compute an initial set of UV-coordinates using spherical following Chapter 4. But since we are interested in learning dense UV-space correspondences across frames consistent with the ground truth UV values, this spherical transformation is done with respect to a learnable scene center instead of a fixed one.

UV-space warp predictor. In our animation model, we learn a basis of UV-space offsets for each surface implemented as an MLP, which receives a set of W learned latent codes and a learned subject code. This defines a set of warp fields that can be linearly combined into a single warp field, which is used to add offsets to the input UV-coordinates before querying the appearance model. In particular, we first *linearly* map the per-frame expression coefficients to a set of weights that *linearly* blend the predicted UV warps across different surfaces. We note that the blending weights are the same for all surfaces, facilitating a lightweight compute at rendering time.

UV-space texture predictor. To account for the geometry and appearance changes that cannot be fully modeled by the UV-offsets (such as the inner mouth, eyelid motions, or complex specularities on the skin), we learn a set of blend-textures in the UV-space that can also be linearly combined into a single layered texture using the expression coefficients. Our texture predictor receives a set of T learned latent codes as well as a learned subject code, and outputs 2-degree spherical harmonics (SH) coefficients for radiance [143] and a scalar alpha value (*i.e.,* we do not estimate volume density as before). Note that since the view-directional radiance is represented via a finite set of SH functions, it is inherently less expressive than the view-directional radiance in Chapter 4. But at the same time, it can be exported as SH coefficient map in the UV-space, allowing quick look-up to render specular effects in a network-free way. Finally, given a UV-space coordinate $\mathbf{u}'$ and a view direction in xyz-space, the output of our entire pipeline is an RGB radiance and alpha value, which are composited across rays to produce the final color:

$$\mathbf{w}_i \triangleq \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \quad \mathbf{c} = \sum_{i=1}^{N} \mathbf{w}_i \mathbf{c_i} \tag{5.6}$$

At inference time, we decompose the radiance into diffuse and specular components, which provides further flexibility on the size of the assets that are exported from the model. In Figure 5.3, we illustrate our renders for each component.

Losses. We train our pipeline in an end-to-end fashion by adopting the following loss function:

$$\mathcal{L} = \mathcal{L}\text{rec} + \lambda_{\text{uv}}\mathcal{L}_{\text{uv}} + \lambda_{\text{silh}}\mathcal{L}_{\text{silh}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} \tag{5.7}$$

where $\mathcal{L}_{\text{rec}}$ is an ℓ_1 reconstruction loss between predicted and ground truth pixel values, $\mathcal{L}_{\text{uv}}$ is an ℓ_1 loss between the per-pixel *expected* UV-coordinates and ground truth UV-values computed only on skin regions of the face, $\mathcal{L}_{\text{silh}}$ is a silhouette loss that guides the manifold geometry using per-image foreground masks [318], and λ_{reg} is the regularization term that penalizes predicted warps, specular radiances as well as manifold predictor weights to promote training stability. Here, the expected UV for each pixel is obtained by a weighted combination of the predicted warped UVs across rays $\{\mathbf{u}'_i\}_{i=1}^N$ as $\bar{\mathbf{u}}' \triangleq \sum_{i=1}^N \mathbf{w}_i\mathbf{u}'_i$, which is supervised to match the ground truth UV-values at that pixel. The UV supervision is a key term to improve the model capacity by registering the facial features consistent with the UV topology of the 3DMM so that the texture basis focuses on appearance changes that cannot be explained by warping of the UV coordinates. We illustrate a representative test frame along with its per-pixel UV values and warps in Figure 5.4.

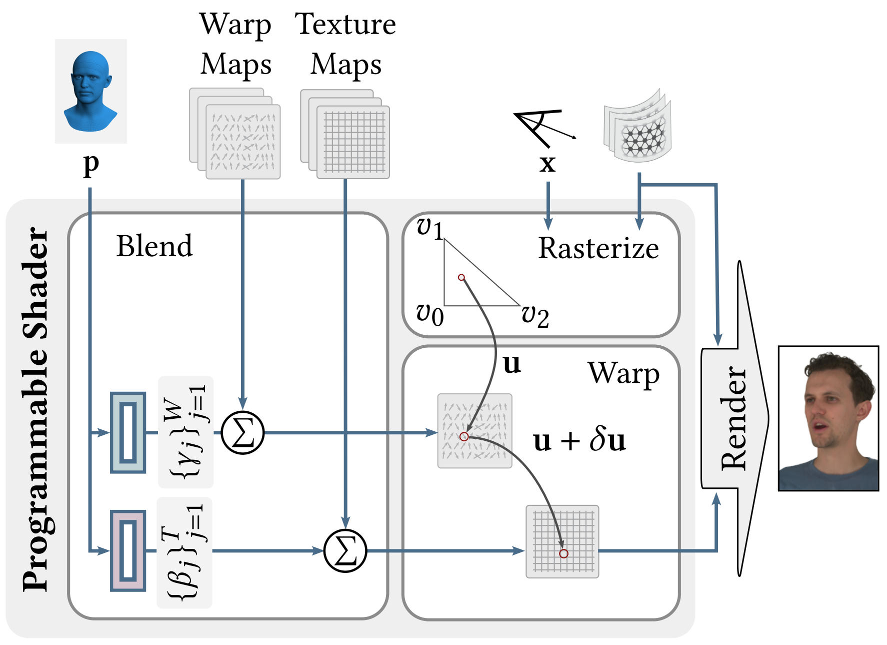

Figure 5.5: **Programmable shader.** At the deployment phase, our 3D assets (a single static layered mesh and bases of warp and texture maps) can easily be used to render dynamic and volumetric faces via a programmable shader on any graphics platform.

Training details. The manifold predictor is implemented as two cascaded 4-layer MLPs of widths 128, which respectively learn a subject-specific 3D warp field and a scalar field that determines the manifold geometry. The warp predictor is a 6-layer MLP of widths 128, which takes in W learnable embeddings of dimension 128 after the third layer. The texture predictor is a 6-layer MLP of widths 256, which receives T learnable embeddings of dimension 128 after the third layer. The subject embeddings are 128-dimensional vectors, which condition each module individually. We optimize our entire model using the Adam optimizer [262] with initial learning rates of 0.0007, 0.0005, 0.0008 and exponential decay rates of 0.20 per 200 000 iterations for $\mathcal{M}$, $\mathcal{W}$, and $\mathcal{T}$, respectively. Using a batch size of 32 768 rays sampled across all subjects, frames, and views, we train our pipeline for 500 000 iterations.

5.3.4 Exporting 3D Assets and Model Deployment

Our discretization logic is similar to the one presented in Chapter 4. We discretize our continuous manifolds into meshes by shooting rays from a hemisphere (centered at the learned scene center) uniformly in azimuth and elevation angles, gathering all intersections and topologizing them into a triangle mesh. We use the same set of points to query our warp and texture predictors to obtain a basis of UV-space offsets and appearance maps. Finally, we export the linear functions that map expression coefficients to blending weights as individual matrices. Depending on the application, the mesh can be decimated to reduce the number of primitives, while UV-offset and appearance maps can be downsampled to lower resolutions. Since our final assets merely comprise a single layered mesh with a fixed topology as well as warp and appearance maps, they can easily be deployed to any graphics platform for rendering.

The blend weights for our warp and texture bases can be efficiently computed at rendering time by linearly mapping $p = 63$ dimensional expression coefficients to $W = T = 12$ coefficients. Including the learned constant offset in this mapping, this results in two weight matrices of size 12×64. Our canonical UV values, warp basis, and texture basis are all exported in the UV space, where the resolution and the precision can be modified for different application needs. Please refer to Figure 5.14 for visualizations of our assets. For renders at 0.5K resolution, we found that mesh, warp map, and texture map resolutions of 512×512 are sufficient to preserve the overall visual quality. Here, a 32-bit precision is maintained for UV values, while the appearance is exported as 8-bit RGBA maps where the view-dependent radiances are discarded.

5.3.5 Rendering

Rendering is performed by a programmable shader that receives the exported 3D assets, as well as face model parameters $\mathbf{p}$ and the camera viewpoint, as shown in Figure 5.5. Here, blend and warp phases are simple linear operations, while rasterization is a standard projection operation handled by the graphics pipeline. By learning a canonical geometry and modeling deformations in the 2D UV-space instead of the 3D space, we can disentangle their parameterization into a single mesh and a set of warp maps. In our results, we show that it is possible to model the blend weights for the warp and appearance bases as a linear transformation of the expression parameters. This makes animation a very simple linear operation without having to explicitly account for complex non-rigid dynamics in 3D. This is in contrast to mesh-based avatar methods such as BakedAvatar [318], which handles animations by explicitly deforming meshes.

A note on rendering efficiency in comparison with modern methods. Rasterizing triangles is inherently significantly faster than the standard implementations of volumetric rendering techniques such as Gaussian splatting and neural fields on a per-primitive and per-pixel basis. Gaussian splatting requires an expensive sorting operation, while neural fields rely on tracing rays, sampling multiple points along the ray that need to be contiguously integrated. Rasterizing our ordered mesh representation does not suffer from such challenges since sorting is handled using the depth buffer. Finally, we *do* note that there are several implementations of 3D Gaussian splatting and neural fields, including hash grids like InstantNGP [141] and hierarchical embeddings [343] that offer faster results, but they are often engineered for particular hardware such as a GPU or require custom implementations for wide-scale deployment.

5.3.6 Streaming

Our method simplifies transmission by initially sending *static* mesh, textures, and linear transformations. Subsequently, only per-frame face model parameters $\mathbf{p}$ are streamed and fed directly to the programmable shader, shown in Figure 5.5. Our technique also offers a unique advantage in *multi-avatar interaction* scenarios. In a 1-on-1 interaction, complete client-side rendering is ideal, as it minimizes the amount of data that must be transmitted. But in a multi-avatar scenario, offloading compute to the server side is more efficient as it avoids duplication of compute effort across multiple clients. Our pipeline allows for expression-related computations such as warping and blending to be performed on the server side, so that only the view-dependent rendering is left to be performed on the client side. Most importantly, since the output of our warping and blending

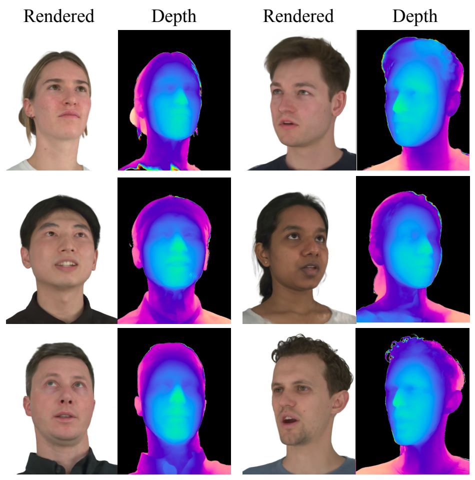

Figure 5.6: **View synthesis results.** Our model achieves photorealistic volumetric rendering of 3D face avatars. Please see the supplementary material in [329] for video demonstrations.

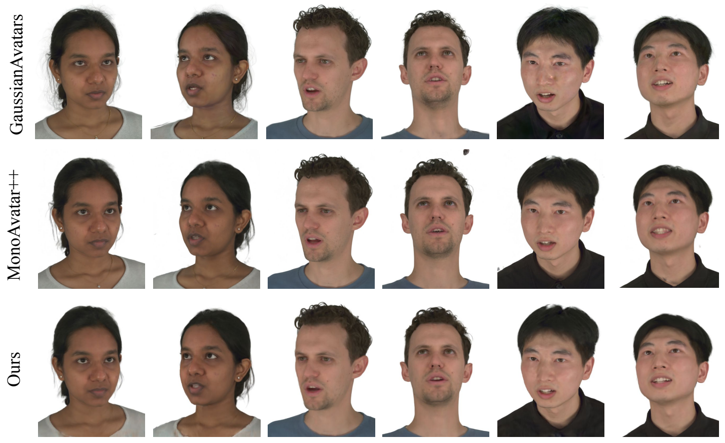

Figure 5.7: **Comparisons on novel view synthesis.** Our model can synthesize novel views at a comparable visual quality to MonoAvatar++ [327] and GaussianAvatars [156], while being less prone to floater artifacts in NeRFs, and inherently preventing primitive ordering artifacts in 3DGS.

operations is a set of texture maps, they can be conveniently transmitted as a compressed video stream. This allows for a healthy trade-off between compute and transmission bandwidth. Such ability is not trivially available with other volumetric techniques, including other mesh-based techniques like BakedAvatar [318], since it demands the much heavier mesh-streaming for every frame. In contrast, our approach relies solely on video streaming.

5.4 Experiments and Results

In our experiments, we set $N = T = W = 12$. We use the first 9 talking sequences of each subject in the NeRSemble dataset [340] for training and use the last one for testing. We hold out 2 of the 16 cameras in training data to evaluate our model's performance quantitatively. To aid the training stability and generalization, we gather 25 subjects from the synthetic data corpus with the closest head shapes to the real subjects, as discussed in Section 5.3.2.

We compare against 5 state-of-the-art efficient volumetric avatar techniques, BakedAvatar [318], Gaussian Head Avatar (GHA) [339], GaussianAvatars [156], Monoavatar++ [327], and PointAvatar [337]. We chose these methods as representative of the best-in-class techniques that achieve fast rendering and employ layered meshes, Gaussian splatting, or neural radiance fields as the underlying volumetric representation. We note that our goal is *not* to visually outperform existing volumetric techniques, but to allow native compatibility with existing graphics platforms, while maintaining comparable visual fidelity.

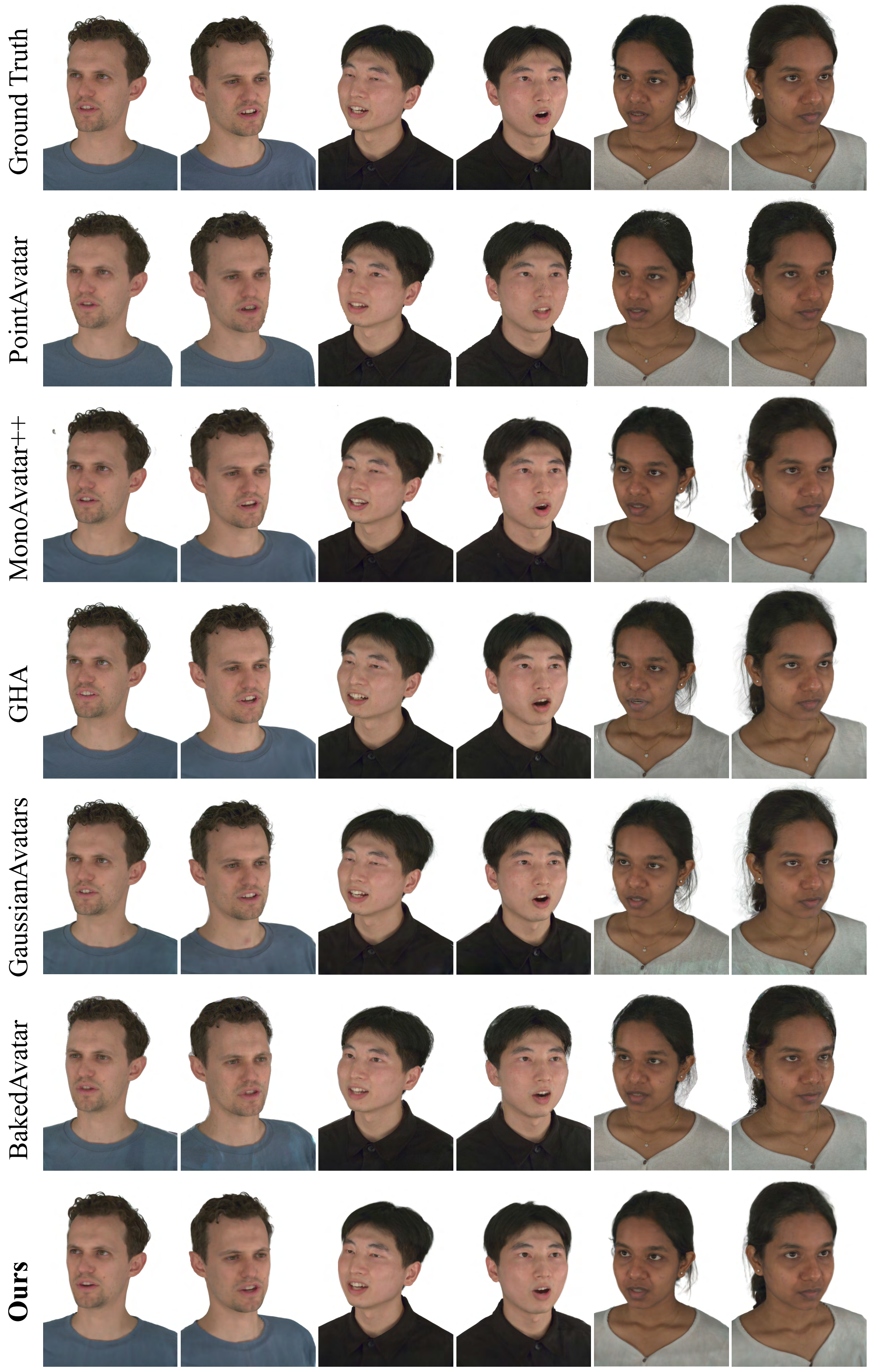

Figure 5.8: **Qualitative comparisons.** Our technique achieves comparable visual quality to modern neural rendering techniques while facilitating 3D animations in a platform-agnostic way.

Table 5.1: **Quantitative comparisons.** Our model attains similar quality compared to previous methods.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
PointAvatar [337]	23.80±1.28	0.872±0.016	0.137±0.018
MonoAvatar++ [327]	27.45±2.43	0.936±0.011	0.098±0.009
GHA [339]	24.29±2.16	0.863±0.039	0.102±0.024
GaussianAvatars [156]	27.54±1.69	0.931±0.017	0.066±0.015
BakedAvatar [318]	24.38±0.78	0.888±0.013	0.117±0.018
Ours	26.97±1.23	0.929±0.007	0.117±0.006

5.4.1 Qualitative Results

Novel view synthesis. We illustrate our novel view synthesis results in Figure 5.6. Our method generalizes over different subjects with varying face geometries and appearances. Please refer to the supplementary video in [329] for demonstrations and comparisons with other methods.

Novel view synthesis comparisons. We provide comparisons on novel view synthesis with the state of the art methods in Figure 5.7. NeRF-based methods like [327] can manifest floating artifacts, and 3DGS-based methods may result in popping-like artifacts due to explicit sorting of primitives [303]. Our method is not prone to such artifacts by design, and the exported textured meshes can be edited by an artist, providing additional flexibility to remove visual seams as a post-processing step. Please see the supplementary video in [329] for better visualizations.

Self-reenactment. We show renders of test expressions from the heldout views and compare them with baseline methods in Figure 5.8. Our model achieves comparable visual quality to the previous methods that rely on sophisticated primitives or MLP queries at rendering time. For video visualizations, please refer to the supplementary material in [329].

Cross-reenactment. Our avatar representation can also be driven by transferring expressions of another subject. Please refer to the supplementary video in [329].

5.4.2 Quantitative Results

For three subjects (with IDs 055, 264, and 306), we perform quantitative evaluations on two held-out views across the entire test sequences, consisting over 800 images. We report average image quality metrics in PSNR, SSIM [273], and LPIPS [274] for our method and other methods in Table 5.1, where we observe comparable average performance.

5.4.3 Real-time Rendering on Web Browsers

Our representation is natively deployable on graphics platforms and enables real-time rendering of volumetric face avatars using a simple programmable shader. Using WebGL on a consumer laptop, we achieve the frame rates shown in Figure 5.9 for varying mesh resolutions and rendering resolutions, while keeping the memory usage less than 2 GB at 512^2 mesh resolution and 2K

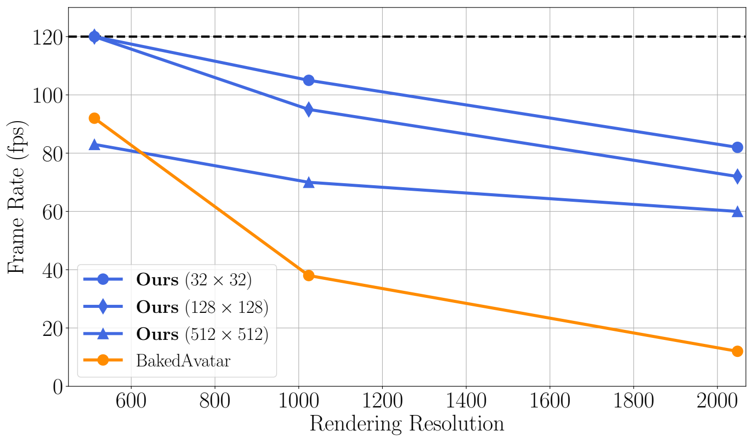

Figure 5.9: **Frame rates on WebGL (in frames per second).** Our assets can be natively deployed on traditional graphical pipelines on web browsers using WebGL, achieving cross-platform compatibility. These numbers are profiled on Chrome 133.0 on a MacBook Pro with M1 Pro chip. Frame rates above the refresh rate of 120 Hz are indicated as >120.

rendering resolution. We emphasize that our approach discretizes all scene components, placing it on the memory-intensive side of the inherent memory–compute trade-off. Nevertheless, by employing a moderate number of layers and basis sizes, our assets remain sufficiently lightweight to maintain a memory footprint compatible with commodity hardware. We also observe that the performance of BakedAvatar [318] suffers significantly at higher rendering resolutions owing to per-pixel MLP queries, while our representation naturally scales well to higher resolutions due to simple texture queries.

5.4.4 Ablation Studies

Mesh and texture resolution. As seen in Chapter 4, our representation has the flexibility to efficiently trade off image quality with rendering efficiency by reducing the number of primitives of the exported mesh and downsampling the layered textures. Given a layered mesh at 512×512 vertex resolution (per layer) with canonical texture coordinates as vertex attributes, we first gather the vertices from each layer as an oriented point cloud and perform Poisson surface reconstruction [103] to obtain watertight surfaces as before. Then, we use an off-the-shelf mesh decimation algorithm to reduce the number of vertices in the mesh to a given target. Since our model is trained to represent a variety of expressions with a single set of static surfaces, the exported meshes roughly correspond to the coarse face geometry of the subjects. Therefore, we can reduce the total number of primitives significantly without sacrificing the visual quality, as shown in Figure 5.10 and the supplementary video in [329].

If there is any need to decrease the resolution of the streamed avatar (such as reduced data bandwidths), we can dynamically downsample our video textures using the existing infrastructure. We report view synthesis and animation results for a variety of texture resolutions, illustrated in Figure 5.10 and the supplementary video in [329].

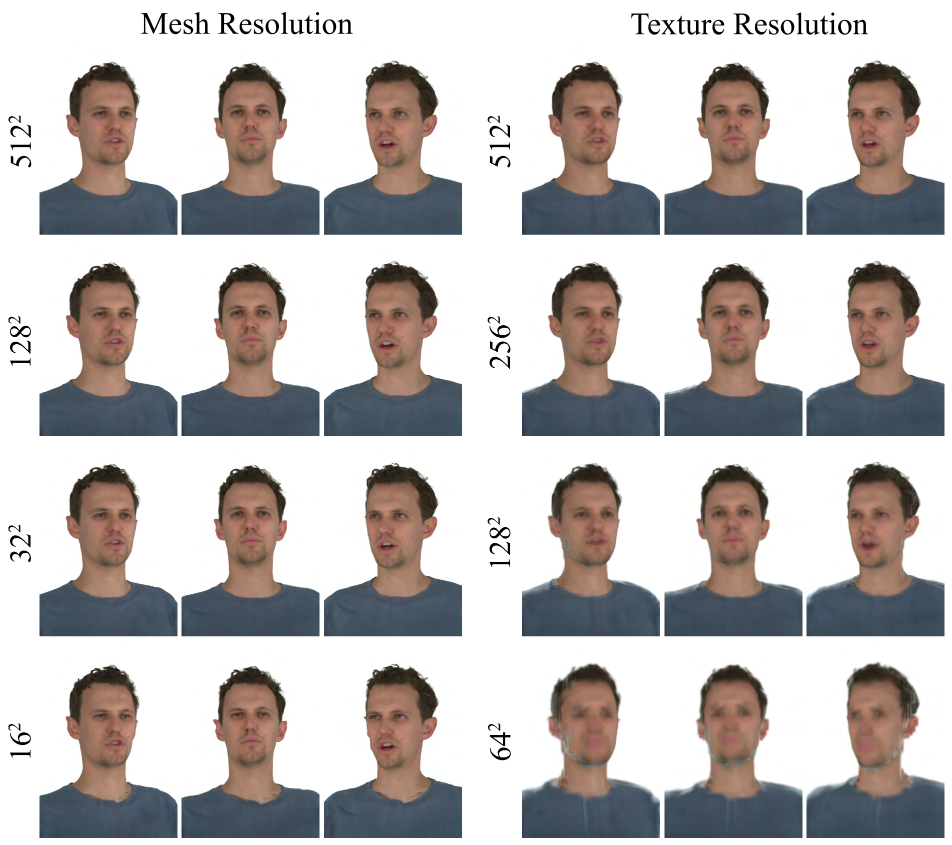

Figure 5.10: **Ablation on mesh and texture resolution.** At deployment phase, our asset size can be trivially reduced with standard operations. Due to our smooth surface geometry, the visual quality is maintained down to 32×32 mesh resolution, with a total number of primitives of <12 000, providing a very lighweight volumetric representation for renders at 0.5K resolution. Similarly, the texture resolution can also be adjusted for varying needs of an application by downsampling layered textures.

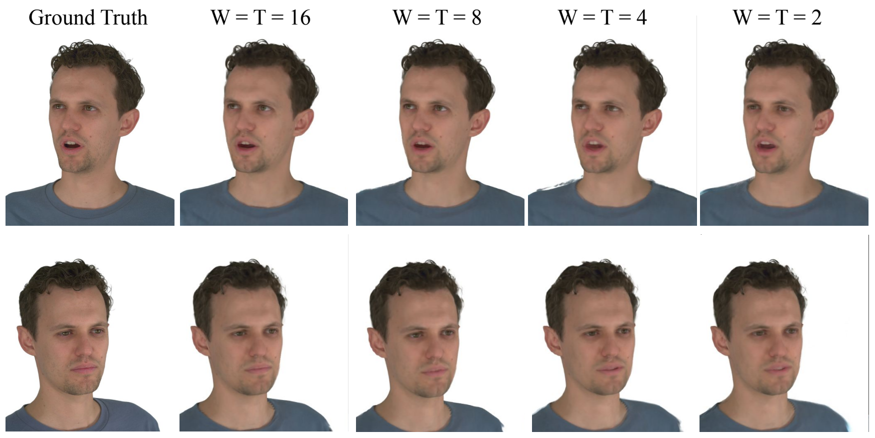

Figure 5.11: **Ablation on sizes of warp and texture bases.** Sufficient number of blendable warps and textures is crucial to achieve good rendering quality and generalization to novel expressions.

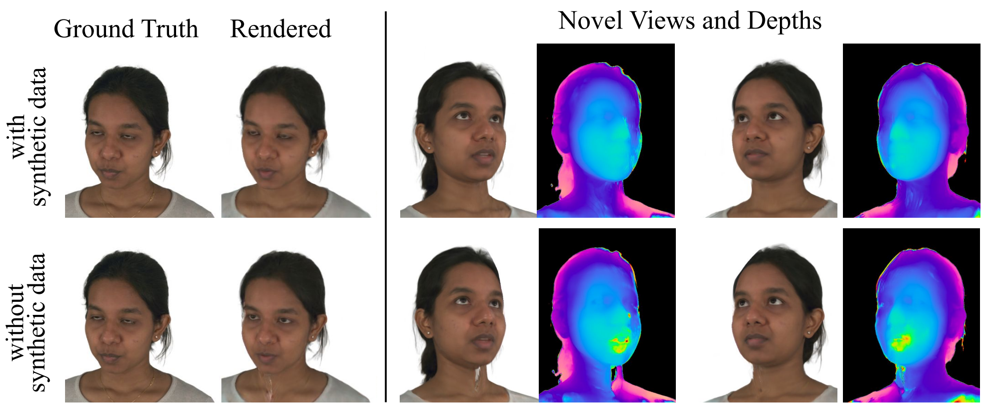

Figure 5.12: **Ablation on synthetic data.** The stability and overfitting challenges can be mitigated by introducing synthetic face data jointly trained with the real subject. This helps with generalization to novel expressions in addition to regularization of the learned face geometry.

Table 5.2: **Ablation study on sizes of warp and texture bases.** Basis sizes improve rendering quality and expression generalization. These metrics are obtained on cropped images that include the face region only.

	PSNR ↑	SSIM ↑	LPIPS ↓
$W = T = 16$	29.67 ± 1.36	0.897 ± 0.012	0.258 ± 0.017
$W = T = 8$	29.47 ± 1.39	0.894 ± 0.012	0.259 ± 0.017
$W = T = 4$	29.38 ± 1.41	0.892 ± 0.012	0.263 ± 0.016
$W = T = 2$	28.46 ± 1.21	0.882 ± 0.012	0.276 ± 0.019

Warp and Texture Basis Size. To provide more insights into our model, we evaluate its expressiveness by modifying its warp and texture basis sizes. We illustrate our results in Figure 5.11 and evaluation metrics in Table 5.2, where we observe that the overall rendering quality suffers and the renders manifest artifacts as we reduce the basis sizes. Furthermore, the model does not generalize to novel facial expressions and eye gazes as we reduce its capacity.

Synthetic data. We illustrate the effectiveness of our joint real–synthetic training in Figure 5.12, where we observe that in the absence of synthetic data, our model is prone to geometric instabilities and may fail to generalize to novel expressions.

5.5 Discussion and Outlook

In this chapter, we introduced a novel, efficient, and natively deployable representation for animatable volumetric head avatars, designed to operate seamlessly within traditional graphics pipelines. We demonstrated that the static mesh and dynamic texture framework developed in the previous chapter can be extended to support expression control, while retaining the practical benefits of memory efficiency, rendering speed, and platform compatibility. This is accomplished by learning bases of UV-space warps and RGBA textures, which together enable the synthesis of a wide range of facial expression changes. Expression control is performed efficiently via

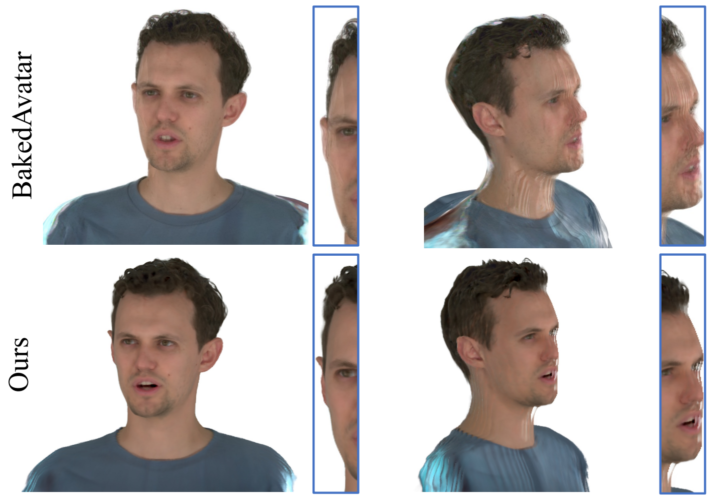

Figure 5.13: **Limitations**. Layered mesh representations may suffer from shell artifacts when viewed from extreme angles. While our approach outperforms existing baselines on visual quality on novel views that deviate from training data *(left)*, at more extremely out-of-training profile views *(right)* it also suffers from "shell" artifacts similar to the baseline.

simple linear blending of these bases, conditioned on tracked parameters from a morphable face model. Our experiments show that discretizing geometry as layered meshes, UV-space warps as pixel-space warp maps, and RGBA textures as pixel-space texture maps supports high-quality rendering with minimal visual artifacts and without incurring significant memory overhead.

Limitations. The core premise of our representation lies in the discretization of scene components: geometry, appearance, and deformation. While this design enables compatibility with traditional rendering pipelines, it inherently imposes limitations on representational capacity when compared to continuous volumetric approaches such as radiance fields or 3D Gaussians. As discussed in the previous chapter, the use of coarse geometry can lead to shell artifacts, particularly at oblique viewing angles, as illustrated in Figure 5.13. Based on our earlier findings, achieving artifact-free rendering at wider viewing angles would require higher-fidelity geometry predictions, which in turn increases the size and complexity of the layered mesh representation.

The enrollment phase in our pipeline depends on machine learning–based training for each subject. While this process can be practically executed using cloud-based computational resources, scaling it to support millions of users poses a significant challenge in terms of cost and efficiency. To address this, recent research has explored strategies to simplify and accelerate the enrollment process. For instance, advances in generative modeling have demonstrated the ability to leverage strong priors over facial geometry and appearance, enabling direct regression of volumetric representations from as little as a single input image [13]. In another line of work, large-scale models trained on massive datasets have shown promise in rapid inference-time enrollment, requiring only a handful of casually captured images from a handheld device [344]. We note that our representation is agnostic to the data modality and thus potentially amenable to training on large-scale, diverse datasets, making it well-suited for integration with these emerging approaches.

In summary, we believe that this work establishes a solid foundation for a novel representation capable of capturing, animating, and rendering volumetric facial effects within conventional graphics frameworks. Future research can build upon this framework to develop even more efficient, scalable, and deployable solutions for real-world applications.

The next era. Early photorealistic head avatar pipelines rely on light stages packed with dozens of synchronized cameras, as standard mesh and texture models lacked enough priors to infer geometry, reflectance, and lighting from a sparse number of views. One breakthrough that resolved these hardware constraints was the combination of neural, differentiable rendering with subject-specific deformations. Nowadays, dynamic NeRF-style volumes (and their 3DGS successors) can be recovered using a single handheld *selfie* video while being warped by a low-dimensional expression and pose codes learned from parametric face models. The next leap appears to be data-hungry pre-training: avatar foundation models can compress millions of subjects into a single transformer that can *cold-start* an enrollment in minutes. Looking a decade ahead, commercial enrollment will likely shrink to a few casually lit photos or a five-second head-turn clip fed through a cloud-scale foundation model. Even if the enrollment converges to a single forward pass, the underlying 3D representation—whether 3D Gaussians, tri-planes, or hybrid mesh and implicit patches—will likely remain a first-order design knob, as it dictates the trade-off between bandwidth, latency, and visual fidelity in production pipelines. In the near future, as devices, their latencies, and privacy constraints will likely remain heterogeneous, it is unlikely that a single avatar foundation model or representation will dominate. Instead, production systems will likely have to juggle a selection of 3D codecs—meshes for legacy, 3D Gaussians for fidelity—and lightweight parametric deformations for expression control, which might involve selecting the one that best satisfies the moment-to-moment trade-off between compute, bandwidth, and photorealism.

The growth of the XR industry, even when equipped with compelling technologies such as virtual telepresence that operate at low cost, low latency, and with high photorealism, must also overcome public hesitation toward large wearable hardware like head-mounted displays. This can imply that these devices must deliver truly groundbreaking capabilities, beyond virtual telepresence alone, to create a strong sense of desire or attraction in users, even when the offered experiences are primarily for entertainment. Therefore, diversifying the application space beyond avatars, with experiences such as immersive gaming, remote collaborative work, or virtual tourism, may accelerate the adoption of these new wearables into our lives.

Looking ahead in the next decade, it is likely that we will witness a new class of transformative technologies emerging, where users are able to generate interactive, fully immersive environments or games from single text prompts. These personalized, shareable *worlds* can offer unprecedented forms of entertainment, with customizable experiences tailored to individual preferences. Interestingly, such virtual environments may entirely deviate from modern-day 3D assets—such as meshes, voxels, point clouds, or radiance fields—and instead be generated directly by large transformer-based architectures that advance beyond current state-of-the-art *world models* such as Genie 3 [345]. Using these models, every pixel displayed on the XR headset could be generated on-the-fly by a transformer, rather than rendered from existing 3D assets. This paradigm opens exciting possibilities for the gaming and entertainment industries, potentially enabling entirely new forms of content creation. For example, an interactive 3D movie generated from a single prompt by a single user could be shared with millions of people, enabling them to experience it on their own devices. However, whether or not these experiences will be sufficient to drive widespread adoption, ultimately leading to a future where XR headsets are as ubiquitous as mobile phones today, remains an open question.

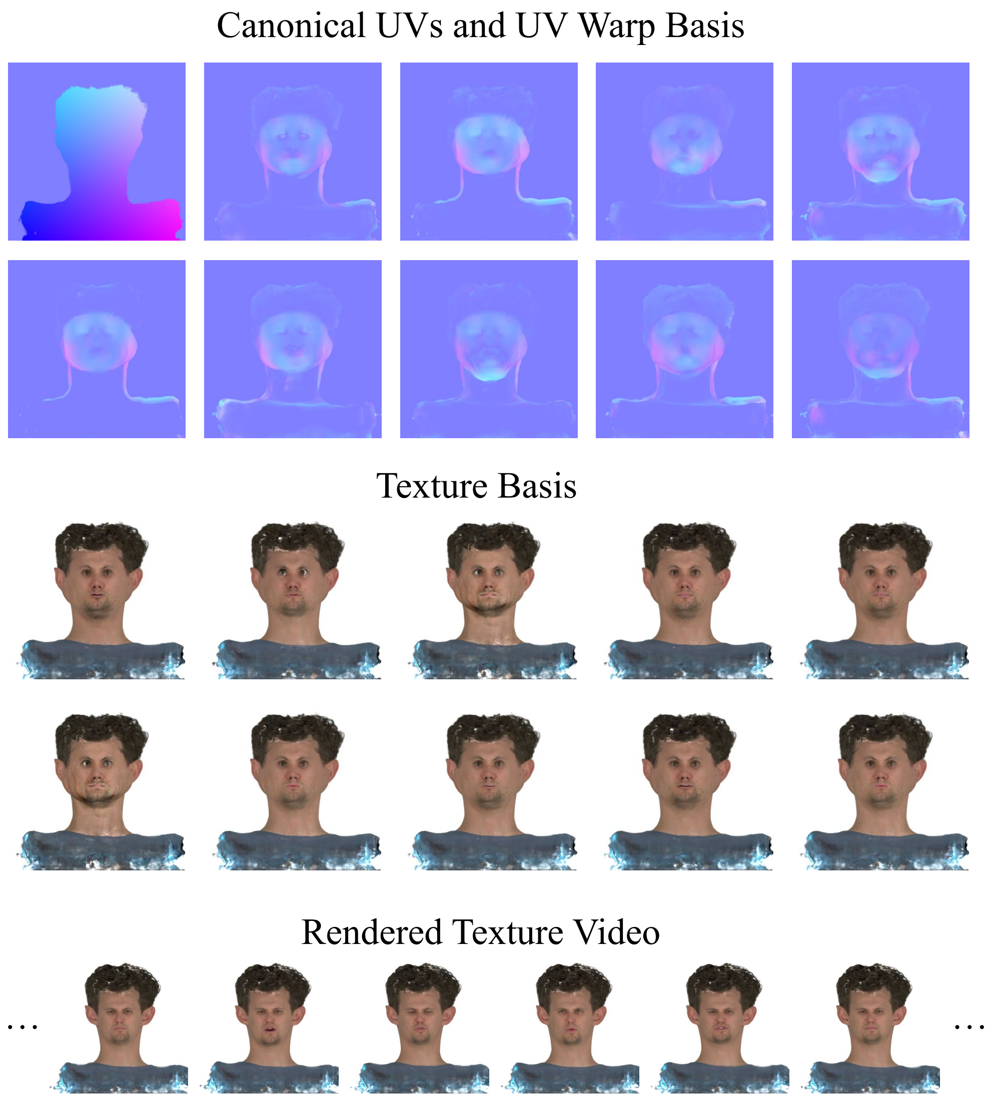

Figure 5.14: **Visualizations of the assets.** Illustrating a subset of the learned warps and appearances. With tracked expression coefficients of a 3DMM, these assets can be used to render a texture video shown at the bottom. All images are alpha-composited for visualization purposes.

6 Concluding Remarks and Perspectives

This thesis offers a narrow yet focused glimpse into the vast and rapidly evolving research landscape surrounding face-centric problems in visual computing. The methods presented were developed over the course of nearly four years, each shaped by a distinct moment in time—situated within vastly different states of computer vision, graphics, and the broader fields of machine learning and artificial intelligence. Over this period, new paradigms have swept through the community, often becoming the de facto state-of-the-art, embraced by thousands of researchers worldwide—only to be replaced by newer frameworks within a year. Generative adversarial networks [205], which formed the foundation of the methods in Chapter 3, have been almost entirely supplanted by diffusion models [89] by the time Chapter 4 was developed. Similarly, large avatar models [344] were not yet on the horizon when the work in Chapter 4 was completed. Today, they are emerging at scale, poised to fundamentally reshape how we think about avatar representations that satisfy a diverse set of design constraints. At the time of writing this thesis, we are witnessing the advent of consumer-accessible reasoning models that bring large language model–level inference capabilities to everyday devices [287]. These systems exhibit unprecedented levels of generalization, rigor, and reasoning ability—capable of addressing research problems that span dozens of disciplines, all within a single interface triggered by a simple user prompt. As of July 2025, an AI model has even been awarded a gold medal in the International Mathematical Olympiad, a symbolic achievement that underscores just how far these technologies have come [346]. Undoubtedly, this may be *the* most exhilarating time to be a researcher in the field of artificial intelligence.

While none of the methods developed in this thesis directly engage with systems that exhibit human-level intelligence through natural language, they nonetheless offer meaningful insights into how machine learning can transform algorithm design, particularly by introducing powerful paradigms that automatically learn rich and fascinating properties of our 3D world, embedded within the weights of neural networks. The central creative challenge lies in unlocking this automation: by selecting the right architecture, curating the appropriate data, designing effective loss functions, and addressing numerous other considerations specific to the 3D vision domain, where algorithms must deal with the complexities of real-world physics, geometry, appearance, illumination, as well as the interactions among them. This makes machine learning in the 3D domain particularly challenging, as the level of abstraction extends beyond simply tokenizing information into embedding vectors. It requires careful modeling of how we digitally represent the 3D world, along with a deep understanding of the image formation process through rendering, while always remaining grounded in physical reality. Indeed, a central theme throughout this thesis has been to critically examine the choice of 3D representation in each task we addressed, and to explore how these representations can be optimized, or even designed from scratch, to better fulfill the requirements of the target applications.

Yet, as we have seen, representation is merely one facet of the broader problem of 3D face synthesis and editing. In Chapter 3, we worked with textured meshes as the foundation for our 3D-aware face image manipulation pipeline. But the ambitious set of desiderata—namely, fast inference, disentangled control, generalization beyond the training distribution, and data efficiency—directly influenced our design choices, ultimately leading to a rather involved, multi-stage training framework. This training scheme required careful and extensive experimentation to balance the relative contributions of multiple loss terms, and to ensure that adversarial supervision consistently provided meaningful gradients to push the synthesized outputs toward better photorealism. In the end, creative model design must often be complemented by thorough empirical study, as the optimal configuration of model architectures or training parameters may not be readily apparent through theoretical reasoning alone.

We observed similar patterns in Chapter 4. While we initially hypothesized that a static layered mesh representation could meet our goals of streamability and backward compatibility, it was not immediately clear whether such a representation would in practice satisfy these constraints, nor how to effectively learn such mesh-based structures. Among various possible approaches, the radiance manifolds framework [307] emerged as a particularly intuitive solution. It enabled the optimization of continuous surfaces that can be rendered directly during training, integrates naturally with gradient-based learning, and responds well to data-driven supervision. However, we also found that the joint learning of geometry and appearance is a highly non-trivial challenge, which often exhibits unstable dynamics as the network attempts to explain variations in one modality using the other. Ironically, our pipeline is explicitly designed to explain most geometric variation through the appearance model. This leads to a spectrum of valid solutions, each placing a different burden on the representational capacity of the geometry and appearance components.

The challenges associated with learning layered mesh representations re-emerged in Chapter 5—this time compounded by additional requirements of controllability and generalization to novel expressions. These new demands required a significant rethinking of the playback pipeline introduced in the previous chapter. Specifically, we introduced UV-space warps to more compactly model geometric deformations, offering a more efficient alternative to directly alpha-compositing fully dynamic textures. Addressing the generalization challenge led us to incorporate synthetic data into the training process, which in turn raised important design questions about subject conditioning and model capacity. To maintain high-quality rendering across a diverse set of identities and expressions, we increased the representational capacity of the network notably. Perhaps the most non-obvious question was whether full discretization of the pipeline, especially the UV-space warps, could preserve the visual fidelity achieved by the continuous formulation. Through empirical validation, we showed that it is indeed possible: a lightweight, programmable shader can consume these discrete assets and drive a volumetric avatar in real time on consumer hardware. This finding holds practical implications for virtual telepresence, potentially enabling scalable deployment across a heterogeneous set of devices, just as modern video conferencing seamlessly spans different web browsers, platforms, and operating systems.

Parting thoughts. A recurring question posed throughout this thesis has been: What are the limitations of the methods we have developed, and what comes next? As we have seen, each of our methods carries a set of trade-offs and limitations that, in turn, spark new directions for addressing the underlying challenges more effectively. At the same time, the broader context in which this thesis was written has been marked by rapid, unprecedented advances in AI and visual computing, leading to paradigm shifts that have already rendered some foundational components of our pipelines less relevant or even obsolete. Pretrained, large-scale diffusion models [285] and vision-language models [289] now provide powerful building blocks for developing face image manipulation systems. Similarly, the emergence of avatar foundation models has begun to eliminate the need for subject-specific, computationally demanding enrollment pipelines altogether. Yet, as we have emphasized throughout this thesis, one of the most pressing open questions remains: What is the *right* 3D representation? This question makes novel representation research a particularly rich and important avenue to pursue. Indeed, a large avatar foundation model that regresses casually captured images into a static layered mesh, along with warp and texture bases, would be a compelling demonstration. However, it is also likely that meshes and textures, long-standing components of traditional graphics pipelines, may soon lose their legacy status. As both academia and industry continue to develop new standards for compression, streaming, and real-time rendering of newer representations, notably radiance fields represented as neural networks or 3D Gaussian primitives, the practical limitations that once hindered these representations will be minimal. When that happens, we can expect a proliferation of consumer-facing applications not just in avatar synthesis, but across the broader spectrum of visual computing tasks, spanning devices from head-mounted AR systems to laptops, smartphones, and potentially a new generation of wearables that may profoundly transform how we interact with the digital world.

The remarkable progress in AI opens up thrilling new possibilities for the future of humanity, but it also brings complex and unsettling uncertainties. Only last year, the feasibility of artificial *general* intelligence was still under debate—now, the conversations have shifted toward the development of artificial *super*intelligence, powered by models that integrate text, audio, images, video, and 3D data. Visual content generation has reached unprecedented levels of realism, where AI-generated videos can now be created by everyday users, shared across social media, and potentially viewed by millions of people. While it is tempting to be swept up in the exciting potentials of these advances, we must again acknowledge a growing divergence: the gap between physical reality and the digitally constructed realities we interact with on our devices is widening. Ultimately, it is up to us, the researchers, not only in computer vision and machine learning but also in the social sciences and humanities, to anticipate and address the potential harms that these technologies may introduce. It is difficult to say whether or not artificial superintelligence will be achieved by the end of this decade. But there is no doubt that the developments in large-scale AI models have begun and will continue to fundamentally change the way we approach and solve problems that involve *synthesizing, editing, and animating 3D faces*, and more broadly, the fields of computer vision and computer graphics, as a whole.

References

[1] S. Zafeiriou, C. Zhang, and Z. Zhang. "A Survey on Face Detection in the Wild: Past, Present and Future". *Computer Vision and Image Understanding* 138 (2015), pp. 1–24. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2015.03.015.

[2] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt. "State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications". *Computer Graphics Forum* 37.2 (2018), pp. 523–550. DOI: https://doi.org/10.1111/cgf.13382.

[3] B. Egger, W. A. P. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, C. Theobalt, V. Blanz, and T. Vetter. "3D Morphable Face Models—Past, Present, and Future". *ACM Trans. Graph.* 39.5 (June 2020). ISSN: 0730-0301. DOI: 10.1145/3395208. URL: https://doi.org/10.1145/3395208.

[4] J. Thevenot, M. B. López, and A. Hadid. "A Survey on Computer Vision for Assistive Medical Diagnosis From Faces". *IEEE Journal of Biomedical and Health Informatics* 22 (2018), pp. 1497–1511.

[5] R. A. Kirsch, L. Cahn, C. Ray, and G. H. Urban. "Experiments in Processing Pictorial Information with a Digital Computer". *Papers and Discussions Presented at the December 9-13, 1957, Eastern Joint Computer Conference: Computers with Deadlines to Meet.* IRE-ACM-AIEE '57 (Eastern). Washington, D.C.: Association for Computing Machinery, 1957, pp. 221–229. ISBN: 9781450378628. DOI: 10.1145/1457720.1457763. URL: https://doi.org/10.1145/1457720.1457763.

[6] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou. "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects". *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (2022), pp. 6999–7019. DOI: 10.1109/TNNLS.2021.3084827.

[7] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. "A Review on Deep Learning Techniques Applied to Semantic Segmentation". *arXiv preprint arXiv:1704.06857* (2017).

[8] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. "Deep Learning for Generic Object Detection: A Survey". 128.2 (Feb. 2020), pp. 261–318. ISSN: 0920-5691. DOI: 10.1007/s11263-019-01247-4. URL: https://doi.org/10.1007/s11263-019-01247-4.

[9] V. Blanz and T. Vetter. “A Morphable Model For The Synthesis Of 3D Faces”. In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400708978. URL: https://doi.org/10.1145/3596711.3596730.

[10] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. “Efficient Geometry-Aware 3D Generative Adversarial Networks”. *ArXiv*. 2021. DOI: 10.1109/CVPR52688.2022.01565.

[11] J. Gu, L. Liu, P. Wang, and C. Theobalt. “StyleNeRF: A Style-based 3D Aware Generator for High-resolution Image Synthesis”. *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=iUuzzTMUw9K.

[12] T. Wang, B. Zhang, T. Zhang, S. Gu, J. Bao, T. Baltrusaitis, J. Shen, D. Chen, F. Wen, Q. Chen, and B. Guo. “RODIN: A Generative Model for Sculpting 3D Digital Avatars Using Diffusion”. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 4563–4573. DOI: 10.1109/CVPR52729.2023.00443.

[13] A. Trevithick, M. Chan, M. Stengel, E. Chan, C. Liu, Z. Yu, S. Khamis, M. Chandraker, R. Ramamoorthi, and K. Nagano. “Real-Time Radiance Fields for Single-Image Portrait View Synthesis”. *ACM Transactions on Graphics (TOG)* (2023). DOI: 10.1145/3592460.

[14] A. R. Bhattarai, M. Nießner, and A. Sevastopolsky. “TriPlaneNet: An Encoder for EG3D Inversion”. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 3055–3065.

[15] J. Sun, X. Wang, Y. Shi, L. Wang, J. Wang, and Y. Liu. “IDE-3D: Interactive Disentangled Editing for High-Resolution 3D-Aware Portrait Synthesis”. *ACM Trans. Graph.* 41.6 (Nov. 2022). ISSN: 0730-0301. DOI: 10.1145/3550454.3555506. URL: https://doi.org/10.1145/3550454.3555506.

[16] C. Kyrlitsias and D. Michael-Grigoriou. “Social Interaction With Agents and Avatars in Immersive Virtual Environments: A Survey”. *Frontiers in Virtual Reality* Volume 2 - 2021 (2022). ISSN: 2673-4192. DOI: 10.3389/frvir.2021.786665.

[17] F. Weidner, G. Boettcher, S. A. Arboleda, C. Diao, L. Sinani, C. Kunert, C. Gerhardt, W. Broll, and A. Raake. “A Systematic Review on the Visualization of Avatars and Agents in AR & VR displayed using Head-Mounted Displays”. *IEEE Transactions on Visualization and Computer Graphics* 29.5 (2023), pp. 2596–2606. DOI: 10.1109/TVCG.2023.3247072.

[18] Y. Deng, J. Yang, D. Chen, F. Wen, and X. Tong. “Disentangled and Controllable Face Image Generation via 3D Imitative-Contrastive Learning”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[19] A. Tewari, M. Elgharib, G. Bharaj, F. Bernard, H.-P. Seidel, P. Perez, M. Zollhöfer, and C. Theobalt. “StyleRig: Rigging StyleGAN for 3D Control Over Portrait Images”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[20] A. Tewari, M. Elgharib, M. B. R, F. Bernard, H.-P. Seidel, P. Pérez, M. Zollhöfer, and C. Theobalt. “PIE: Portrait Image Embedding for Semantic Control”. *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: 10.1145/3414685.3417803. URL: https://doi.org/10.1145/3414685.3417803.

[21] J.-y. Noh and U. Neumann. *A Survey of Facial Modeling and Animation Techniques.* Tech. rep. USC Technical Report, 99–705, 1998.

[22] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. "FaceVR: Real-Time Gaze-Aware Facial Reenactment in Virtual Reality". *ACM Trans. Graph.* 37.2 (June 2018). ISSN: 0730-0301. DOI: 10.1145/3182644. URL: https://doi.org/10.1145/3182644.

[23] H. Chu, S. Ma, F. De la Torre, S. Fidler, and Y. Sheikh. "Expressive Telepresence via Modular Codec Avatars". *European Conference on Computer Vision.* Springer. 2020, pp. 330–345.

[24] H. E. Sumbul, T. F. Wu, Y. Li, S. S. Sarwar, W. Koven, E. Murphy-Trotzky, X. Cai, E. Ansari, D. H. Morris, H. Liu, D. Kim, E. Beigne, R. Labs, and Meta. "System-Level Design and Integration of a Prototype AR/VR Hardware Featuring a Custom Low-Power DNN Accelerator Chip in 7nm Technology for Codec Avatars". *2022 IEEE Custom Integrated Circuits Conference (CICC).* 2022, pp. 01–08. DOI: 10.1109/CICC53496.2022.9772810.

[25] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms". *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06).* Vol. 1. 2006, pp. 519–528. DOI: 10.1109/CVPR.2006.19.

[26] O. Özyeşil, V. Voroninski, R. Basri, and A. Singer. "A Survey of Structure from Motion". *Acta Numerica* 26 (2017), pp. 305–364.

[27] X.-F. Han, H. Laga, and M. Bennamoun. " Image-Based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era ". *IEEE Transactions on Pattern Analysis & Machine Intelligence* 43.05 (May 2021), pp. 1578–1604. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2019.2954885.

[28] B. Wang, L. Jiang, J. Li, H. Cai, and H. Liu. "Grasping Unknown Objects Based on 3D Model Reconstruction". *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.* IEEE. 2005, pp. 461–466.

[29] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

[30] M. Sra, S. Garrido-Jurado, C. Schmandt, and P. Maes. "Procedurally Generated Virtual Reality from 3D Reconstructed Physical Space". *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology.* 2016, pp. 191–200.

[31] S. Orts-Escolano et al. "Holoportation: Virtual 3D Teleportation in Real-Time". *Proceedings of the 29th Annual Symposium on User Interface Software and Technology.* ACM, Oct. 2016. DOI: 10.1145/2984511.2984517.

[32] M.-D. Yang, C.-F. Chao, K.-S. Huang, L.-Y. Lu, and Y.-P. Chen. "Image-based 3D Scene Reconstruction and Exploration in Augmented Reality". *Automation in Construction* 33 (2013), pp. 48–60.

[33] R. W. Brown, Y.-C. N. Cheng, E. M. Haacke, M. R. Thompson, and R. Venkatesan. *Magnetic Resonance Imaging: Physical Principles and Sequence Design.* John Wiley & Sons, 2014.

[34] J. Hsieh. *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*. 3rd. SPIE, 2015, p. 574.

[35] T. Würfl, F. C. Ghesu, V. Christlein, and A. Maier. "Deep Learning Computed Tomography". *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. 2016, pp. 432–440.

[36] L. Li, N. Schemenauer, X. Peng, Y. Zeng, and P. Gu. "A Reverse Engineering System for Rapid Manufacturing of Complex Objects". *Robotics and Computer-Integrated Manufacturing* 18.1 (2002), pp. 53–67.

[37] X. Ye, H. Liu, L. Chen, Z. Chen, X. Pan, and S. Zhang. "Reverse Innovative Design—an Integrated Product Design Methodology". *Computer-Aided design* 40.7 (2008), pp. 812–827.

[38] L. Gomes, O. R. P. Bellon, and L. Silva. "3D Reconstruction Methods for Digital Preservation of Cultural Heritage: A Survey". *Pattern Recognition Letters* 50 (2014), pp. 3–14.

[39] R. Di Maio, M. La Manna, and E. Piegari. "3D Reconstruction of Buried Structures from Magnetic, Electromagnetic and ERT Data: Example from the Archaeological Site of Phaistos (Crete, Greece)". *Archaeological Prospection* 23.1 (2016), pp. 3–13.

[40] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. "Nerfies: Deformable Neural Radiance Fields". *ICCV* (2021). DOI: 10.1109/ICCV48922.2021.00581.

[41] K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz. "HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields". *ACM Trans. Graph* 40.6 (Dec. 2021). DOI: 10.1145/3478513.3480487.

[42] Z. Yang, X. Gao, W. Zhou, S. Jiao, Y. Zhang, and X. Jin. "Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2024, pp. 20331–20341.

[43] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. USA: Society for Industrial and Applied Mathematics, 2004. ISBN: 0898715725.

[44] M. Bertero, P. Boccacci, and C. De Mol. *Introduction to Inverse Problems in Imaging*. CRC Press, 2021.

[45] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Nature, 2022.

[46] A. Torralba, P. Isola, and W. Freeman. *Foundations of Computer Vision*. Adaptive Computation and Machine Learning series. MIT Press, 2024. ISBN: 9780262378666. URL: https://mitpress.mit.edu/9780262048972/foundations-of-computer-vision/.

[47] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. *Polygon Mesh Processing*. CRC press, 2010.

[48] M. Jones, J. Baerentzen, and M. Sramek. "3D Distance Fields: A Survey of Techniques and Applications". *IEEE Transactions on Visualization and Computer Graphics* 12.4 (2006), pp. 581–599. DOI: 10.1109/TVCG.2006.56.

[49] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". *Commun. ACM* 65.1 (Dec. 2021), pp. 99–106. ISSN: 0001-0782. DOI: 10.1145/3503250. URL: https://doi.org/10.1145/3503250.

[50] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". *ACM Transactions on Graphics* 42.4 (July 2023). DOI: 10.1145/3592433. URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[51] F. Natterer. *The Mathematics of Computerized Tomography*. Society for Industrial and Applied Mathematics, 2001. DOI: 10.1137/1.9780898719284. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898719284. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898719284.

[52] C. R. Vogel. *Computational Methods for Inverse Problems*. Society for Industrial and Applied Mathematics, 2002. DOI: 10.1137/1.9780898717570. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898717570. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898717570.

[53] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. "Solving Inverse Problems using Data-driven Models". *Acta Numerica* 28 (2019), pp. 1–174.

[54] P. R. Walker. *The Feud that Sparked the Renaissance: How Brunelleschi and Ghiberti Changed the Art World*. Harper Collins, 2009.

[55] A. Pozzo. *Perspectiva pictorum et architectorum*. Latin and Italian. Romae: Typis Joannis Jacobi Komarek, 1693. URL: https://archive.org/details/gri_33125008639367.

[56] P. Campisi and K. Egiazarian. *Blind Image Deconvolution: Theory and Applications*. CRC press, 2017.

[57] M. Elad, B. Kawar, and G. Vaksman. "Image Denoising: The Deep Learning Revolution and Beyond—a Survey Paper". *SIAM Journal on Imaging Sciences* 16.3 (2023), pp. 1594–1654.

[58] Z. Wang, J. Chen, and S. C. Hoi. "Deep Learning for Image Super-resolution: A Survey". *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[59] X. Li, B. Gunturk, and L. Zhang. "Image Demosaicing: A Systematic Survey". *Visual Communications and Image Processing 2008*. Vol. 6822. SPIE. 2008, pp. 489–503.

[60] H. Laga, L. V. Jospin, F. Boussaid, and M. Bennamoun. "A Survey on Deep Learning Techniques for Stereo-Based Depth Estimation". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2022), pp. 1738–1764. DOI: 10.1109/TPAMI.2020.3032602.

[61] G. Patow and X. Pueyo. "A Survey of Inverse Rendering Problems". *Computer Graphics Forum*. Vol. 22. 4. Wiley Online Library. 2003, pp. 663–687.

[62] G. Ongie, A. Jalal, C. A. Metzler, R. Baraniuk, A. Dimakis, and R. Willett. "Deep Learning Techniques for Inverse Problems in Imaging". *IEEE Journal on Selected Areas in Information Theory* 1 (2020), pp. 39–56. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=9084378.

[63] H. W. Engl and R. Ramlau. "Regularization of Inverse Problems". In: *Encyclopedia of Applied and Computational Mathematics*. Springer, 2015, pp. 1233–1241.

[64] B. Jähne. *Digital Image Processing.* 6th ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-24035-8. DOI: 10.1007/3-540-27563-0.

[65] J. Nakamura. *Image Sensors and Signal Processing for Digital Still Cameras.* USA: CRC Press, Inc., 2005. ISBN: 0849335450.

[66] P. Domingos. “The Role of Occam’s Razor in Knowledge Discovery”. *Data Mining and Knowledge Discovery* 3.4 (1999), pp. 409–425.

[67] A. Hyvärinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.* Vol. 39. Springer Science & Business Media, 2009.

[68] D. Ruderman and W. Bialek. “Statistics of natural images: Scaling in the woods”. *Advances in Neural Information Processing Systems* (1993).

[69] S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way.* 3rd. USA: Academic Press, Inc., 2008. ISBN: 0123743702.

[70] E. J. Candès and D. L. Donoho. *Curvelets: A Surprisingly Effective Nonadaptive Representation for Objects with Edges.* Tech. rep. Technical Report 1999-28. Stanford, CA: Department of Statistics, Stanford University, 1999.

[71] D. Strong and T. Chan. “Edge-Preserving and Scale-Dependent Properties of Total Variation Regularization”. *Inverse Problems* 19.6 (Nov. 2003), S165. DOI: 10.1088/0266-5611/19/6/059. URL: https://dx.doi.org/10.1088/0266-5611/19/6/059.

[72] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. “Laplacian Surface Editing”. *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing.* 2004, pp. 175–184.

[73] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. “Deep Convolutional Neural Network for Inverse Problems in Imaging”. *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522.

[74] J. C. Ye, Y. Han, and E. Cha. “Deep Convolutional Framelets: A general Deep Learning Framework for Inverse Problems”. *SIAM Journal on Imaging Sciences* 11.2 (2018), pp. 991–1048.

[75] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. *Advances in Neural Information Processing Systems* (2012).

[76] L. Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. *Proceedings of COMPSTAT’2010.* Ed. by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186. ISBN: 978-3-7908-2604-3.

[77] A. Krogh and J. Hertz. “A Simple Weight Decay can Improve Generalization”. *Advances in Neural Information Processing Systems* (1991).

[78] A. Y. Ng. “Feature selection, L1 vs. L2 regularization, and rotational invariance”. *Proceedings of the Twenty-First International Conference on Machine Learning.* ICML ’04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 78. ISBN: 1581138385. DOI: 10.1145/1015330.1015435. URL: https://doi.org/10.1145/1015330.1015435.

[79] N. Cohen and A. Shashua. “Inductive Bias of Deep Convolutional Networks through Pooling Geometry”. *International Conference on Learning Representations.* 2017. URL: https://openreview.net/forum?id=BkVsEMYel.

[80] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Deep Image Prior”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 9446–9454.

[81] V. Monga, Y. Li, and Y. C. Eldar. “Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing”. *IEEE Signal Processing Magazine* 38.2 (2021), pp. 18–44.

[82] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. “Deep Convolutional Inverse Graphics Network”. *Advances in Neural Information Processing Systems* (2015).

[83] Z. Liu, P. Luo, X. Wang, and X. Tang. “Deep Learning Face Attributes in the Wild”. *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV).* ICCV ’15. USA: IEEE Computer Society, 2015, pp. 3730–3738. ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.425. URL: https://doi.org/10.1109/ICCV.2015.425.

[84] T. Karras, S. Laine, and T. Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019. DOI: 10.1109/TPAMI.2020.2970919.

[85] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. “StarGAN v2: Diverse Image Synthesis for Multiple Domains”. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2020, pp. 8188–8197.

[86] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. “3D Object Representations for Fine-Grained Categorization”. *2013 IEEE International Conference on Computer Vision Workshops.* 2013, pp. 554–561. DOI: 10.1109/ICCVW.2013.77.

[87] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. “LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop”. *arXiv preprint arXiv:1506.03365* (2015).

[88] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. *Proceedings of the 32nd International Conference on Machine Learning.* Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265. URL: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[89] J. Ho, A. Jain, and P. Abbeel. “Denoising Diffusion Probabilistic Models”. *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[90] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. “Score-Based Generative Modeling through Stochastic Differential Equations”. *International Conference on Learning Representations.* 2021. URL: https://openreview.net/forum?id=PxTIG12RRHS.

[91] G. Daras, H. Chung, C.-H. Lai, Y. Mitsufuji, J. C. Ye, P. Milanfar, A. G. Dimakis, and M. Delbracio. “A Survey on Diffusion Models for Inverse Problems”. *arXiv preprint arXiv:2410.00083* (2024).

[92] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. Van Gool. “Denoising Diffusion Models for Plug-and-Play Image Restoration”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2023, pp. 1219–1229.

[93] M. Mardani, J. Song, J. Kautz, and A. Vahdat. "A Variational Perspective on Solving Inverse Problems with Diffusion Models". *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=1YO4EE3SPB.

[94] Z. Wu, Y. Sun, Y. Chen, B. Zhang, Y. Yue, and K. Bouman. "Principled Probabilistic Imaging using Diffusion Models as Plug-and-Play Priors". *Advances in Neural Information Processing Systems* (2024), pp. 118389–118427.

[95] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. "Image Super-Resolution via Iterative Refinement". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2023), pp. 4713–4726. DOI: 10.1109/TPAMI.2022.3204461.

[96] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, and M. Norouzi. "Palette: Image-to-Image Diffusion Models". *ACM SIGGRAPH 2022 Conference Proceedings*. SIGGRAPH '22. Vancouver, BC, Canada: Association for Computing Machinery, 2022. ISBN: 9781450393379. DOI: 10.1145/3528233.3530757. URL: https://doi.org/10.1145/3528233.3530757.

[97] G. Luo, M. Blumenthal, M. Heide, and M. Uecker. "Bayesian MRI Reconstruction with Joint Uncertainty Estimation using Diffusion Models". *Magnetic Resonance in Medicine* 90.1 (2023), pp. 295–311.

[98] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. "DreamFusion: Text-to-3D using 2D Diffusion". *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=FjNys5c7VyY.

[99] A. Tewari, T. Yin, G. Cazenavette, S. Rezchikov, J. Tenenbaum, F. Durand, B. Freeman, and V. Sitzmann. "Diffusion with Forward Models: Solving Stochastic Inverse Problems without Direct Supervision". *Advances in Neural Information Processing Systems* 36 (2023), pp. 12349–12362.

[100] R. Lukac and K. Plataniotis. "Color Filter Arrays: Design and Performance Analysis". *IEEE Transactions on Consumer Electronics* 51.4 (2005), pp. 1260–1267. DOI: 10.1109/TCE.2005.1561853.

[101] J. Shan and C. K. Toth. *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, 2018.

[102] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Surface Reconstruction from Unorganized Points". SIGGRAPH '92. New York, NY, USA: Association for Computing Machinery, 1992, pp. 71–78. ISBN: 0897914791. DOI: 10.1145/133994.134011. URL: https://doi.org/10.1145/133994.134011.

[103] M. Kazhdan, M. Bolitho, and H. Hoppe. "Poisson Surface Reconstruction". *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. Vol. 7. 2006, p. 0. DOI: 10.1145/2487228.2487237x26amp.

[104] S. Laine and T. Karras. "Efficient Sparse Voxel Octrees". *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. Washington, D.C.: Association for Computing Machinery, 2010, pp. 55–63. ISBN: 9781605589398. DOI: 10.1145/1730804.1730814. URL: https://doi.org/10.1145/1730804.1730814.

[105] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. “Real-time 3D reconstruction at scale using voxel hashing”. *ACM Trans. Graph.* 32.6 (Nov. 2013). ISSN: 0730-0301. DOI: 10.1145/2508363.2508374. URL: https://doi.org/10.1145/2508363.2508374.

[106] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 165–174.

[107] C. de Boor. *A Practical Guide to Splines.* Revised. Vol. 27. Applied Mathematical Sciences. New York: Springer, 2001. ISBN: 978-0-387-95366-3. DOI: 10.1007/978-0-387-21779-6.

[108] L. A. Piegl and W. Tiller. *The NURBS Book.* 2nd ed. Berlin, Heidelberg: Springer, 1997. ISBN: 978-3-540-61545-3. DOI: 10.1007/978-3-642-59223-2.

[109] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. “Multiresolution analysis of arbitrary meshes”. *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH ’95. New York, NY, USA: Association for Computing Machinery, 1995, pp. 173–182. ISBN: 0897917014. DOI: 10.1145/218380.218440. URL: https://doi.org/10.1145/218380.218440.

[110] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 4460–4470.

[111] D. Vicini, S. Speierer, and W. Jakob. “Differentiable Signed Distance Function Rendering”. *ACM Trans. Graph.* 41.4 (July 2022). ISSN: 0730-0301. DOI: 10.1145/3528223.3530139. URL: https://doi.org/10.1145/3528223.3530139.

[112] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. “Implicit Geometric Regularization for Learning Shapes”. *ArXiv Preprint ArXiv:2002.10099* (2020). URL: https://arxiv.org/pdf/2002.10099.

[113] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. “NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction”. *Proceedings of the 35th International Conference on Neural Information Processing Systems.* NIPS ’21. Red Hook, NY, USA: Curran Associates Inc., 2021. ISBN: 9781713845393.

[114] Y. Chen and X. Wang. “Transformers as Meta-learners for Implicit Neural Representations”. *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII.* Tel Aviv, Israel: Springer-Verlag, 2022, pp. 170–187. ISBN: 978-3-031-19789-5. DOI: 10.1007/978-3-031-19790-1_11. URL: https://doi.org/10.1007/978-3-031-19790-1_11.

[115] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin. “Neuralangelo: High-Fidelity Neural Surface Reconstruction”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2023, pp. 8456–8465. DOI: 10.1109/CVPR52729.2023.00817.

[116] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering.* 4th ed. Boca Raton, FL: A K Peters/CRC Press, 2018. ISBN: 978-1-138-48498-9.

[117] G. Fyffe, T. Hawkins, C. Watts, W.-C. Ma, and P. Debevec. "Comprehensive Facial Performance Capture". *Computer Graphics Forum* 30.2 (2011), pp. 425–434. DOI: 10.1111/j.1467-8659.2011.01888.x.

[118] P. Gotardo, J. Riviere, D. Bradley, A. Ghosh, and T. Beeler. "Practical Dynamic Facial Appearance Modeling and Acquisition". *ACM Trans. Graph* 37.6 (Dec. 2018). ISSN: 0730-0301. DOI: 10.1145/3272127.3275073. URL: https://doi.org/10.1145/3272127.3275073.

[119] K. Guo et al. "The relightables: volumetric performance capture of humans with realistic relighting". *ACM Trans. Graph.* 38.6 (Nov. 2019). ISSN: 0730-0301. DOI: 10.1145/3355089.3356571. URL: https://doi.org/10.1145/3355089.3356571.

[120] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. "High-Quality Streamable Free-Viewpoint Video". *ACM Trans. Graph* 34.4 (July 2015). ISSN: 0730-0301. DOI: 10.1145/2766945. URL: https://doi.org/10.1145/2766945.

[121] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. R. Salama, and D. Weiskopf. "Real-time Volume Graphics". *ACM SIGGRAPH 2004 Course Notes.* SIGGRAPH '04. Los Angeles, CA: Association for Computing Machinery, 2004, 29–es. ISBN: 9781450378017. DOI: 10.1145/1103900.1103929. URL: https://doi.org/10.1145/1103900.1103929.

[122] S. Seitz and C. Dyer. "Photorealistic Scene Reconstruction by Voxel Coloring" (1997), pp. 1067–1073. DOI: 10.1109/CVPR.1997.609462.

[123] J. S. De Bonet and P. Viola. "Poxels: Probabilistic Voxelized Volume Reconstruction". *Proceedings of International Conference on Computer Vision (ICCV).* Vol. 2. 1999, p. 2.

[124] K. Kutulakos and S. Seitz. "A Theory of Shape by Space Carving". 1 (1999), 307–314 vol.1. DOI: 10.1109/ICCV.1999.791235.

[125] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". *European Conference on Computer Vision.* Springer. 2016, pp. 628–644.

[126] G. Riegler, A. Osman Ulusoy, and A. Geiger. "OctNet: Learning Deep 3D Representations at High Resolutions". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2017, pp. 3577–3586.

[127] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. "Neural Volumes: Learning Dynamic Renderable Volumes From Images". *ACM Trans. Graph* 38.4 (2019), 65:1–65:14. DOI: 10.1145/3306346.3323020.

[128] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollöfer. "DeepVoxels: Learning Persistent 3D Feature Embeddings". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 2437–2446.

[129] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt. "Neural Sparse Voxel Fields". *Advances in Neural Information Processing Systems* 33 (2020), pp. 15651–15663.

[130] M. Levoy and P. Hanrahan. "Light Field Rendering". *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, pp. 31–42. ISBN: 0897917464. DOI: 10.1145/237170.237199. URL: https://doi.org/10.1145/237170.237199.

[131] J. T. Kajiya and B. P. Von Herzen. "Ray Tracing Volume Densities". SIGGRAPH '84. New York, NY, USA: Association for Computing Machinery, 1984, pp. 165–174. ISBN: 0897911385. DOI: 10.1145/800031.808594. URL: https://doi.org/10.1145/800031.808594.

[132] A. Tagliasacchi and B. Mildenhall. "Volume Rendering Digest (for NeRF)". *arXiv preprint arXiv:2209.02417* (2022).

[133] N. Max. "Optical Models for Direct Volume Rendering". *IEEE Transactions on Visualization and Computer Graphics* 1.2 (2002), pp. 99–108.

[134] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, et al. "Advances in Neural Rendering". *Computer Graphics Forum*. Vol. 41. 2. Wiley Online Library. 2022, pp. 703–735.

[135] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li. "NeRF: Neural Radiance Field in 3D Vision: a Comprehensive Review". *arXiv preprint arXiv:2210.00379* (2022).

[136] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. "Neural Fields in Visual Computing and Beyond". *Computer Graphics Forum*. Vol. 41. 2. Wiley Online Library. 2022, pp. 641–676.

[137] H. Lin, S. Peng, Z. Xu, Y. Yan, Q. Shuai, H. Bao, and X. Zhou. "Efficient Neural Radiance Fields for Interactive Free-Viewpoint Video". *SIGGRAPH Asia 2022 Conference Papers*. 2022, pp. 1–9.

[138] Y. Hong, B. Peng, H. Xiao, L. Liu, and J. Zhang. "HeadNeRF: A Real-Time NeRF-Based Parametric Head Model". *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. DOI: 10.1109/CVPR52688.2022.01973.

[139] Z. Bai, F. Tan, Z. Huang, K. Sarkar, D. Tang, D. Qiu, A. Meka, R. Du, M. Dou, S. Orts-Escolano, R. Pandey, P. Tan, T. Beeler, S. Fanello, and Y. Zhang. "Learning Personalized High Quality Volumetric Head Avatars From Monocular RGB Videos". *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023. DOI: 10.1109/CVPR52729.2023.01620.

[140] T. Li, M. Slavcheva, M. Zollhoefer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, R. Newcombe, et al. "Neural 3D Video Synthesis from Multi-View Video". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5521–5531.

[141] T. Müller, A. Evans, C. Schied, and A. Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[142] Sara Fridovich-Keil and Alex Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. "Plenoxels: Radiance Fields Without Neural Networks". *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022. DOI: 10.1109/CVPR52688.2022.00542.

[143] R. Ramamoorthi and P. Hanrahan. "An Efficient Representation for Irradiance Environment Maps". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. 2001, pp. 497–500.

[144] C. Lassner and M. Zollhofer. "Pulsar: Efficient Sphere-based Neural Rendering". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1440–1449.

[145] M. Gross and H. Pfister. *Point-Based Graphics*. Elsevier, 2011.

[146] S. Rusinkiewicz and M. Levoy. "QSplat: A Multiresolution Point Rendering System for Large Meshes". *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 343–352. ISBN: 1581132085. DOI: 10.1145/344779.344940. URL: https://doi.org/10.1145/344779.344940.

[147] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. "Surfels: Surface Elements as Rendering Primitives". *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 335–342. ISBN: 1581132085. DOI: 10.1145/344779.344936. URL: https://doi.org/10.1145/344779.344936.

[148] M. Botsch, A. Hornung, M. Zwicker, and L. Kobbelt. "High-quality Surface Splatting on Today's GPUs". *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* IEEE. 2005, pp. 17–141.

[149] C. Müller. *Spherical Harmonics*. Vol. 17. Springer, 2006.

[150] R. Green. "Spherical Harmonic Lighting: The Gritty Details". *Archives of the Game Developers Conference*. Vol. 56. 2003, p. 4.

[151] N. Snavely, S. M. Seitz, and R. Szeliski. "Photo Tourism: Exploring Photo Collections in 3D". In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 1st ed. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400708978. URL: https://doi.org/10.1145/3596711.3596766.

[152] S. Rota Bulò, L. Porzi, and P. Kontschieder. "Revising Densification in Gaussian Splatting". *European Conference on Computer Vision*. Springer. 2024, pp. 347–362.

[153] Y. Bao, T. Ding, J. Huo, Y. Liu, Y. Li, W. Li, Y. Gao, and J. Luo. "3D Gaussian Splatting: Survey, Technologies, Challenges, and Opportunities". *IEEE Transactions on Circuits and Systems for Video Technology* (2025).

[154] T. Wu, Y.-J. Yuan, L.-X. Zhang, J. Yang, Y.-P. Cao, L.-Q. Yan, and L. Gao. "Recent Advances in 3D Gaussian Splatting". *Computational Visual Media* 10.4 (2024), pp. 613–642.

[155] Y. Liang, N. Khan, Z. Li, T. Nguyen-Phuoc, D. Lanman, J. Tompkin, and L. Xiao. "GauFRe: Gaussian Deformation Fields for Real-time Dynamic Novel View Synthesis". *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2025, pp. 2642–2652.

[156] S. Qian, T. Kirschstein, L. Schoneveld, D. Davoli, S. Giebenhain, and M. Nießner. "GaussianAvatars: Photorealistic Head Avatars with Rigged 3D Gaussians". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20299–20309.

[157] M. T. Bagdasarian, P. Knoll, Y. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern. "3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods". *Computer Graphics Forum*. Wiley Online Library. 2024, e70078.

[158] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, Z. Wang, et al. "LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS". *Advances in Neural Information Processing Systems* (2024), pp. 140138–140158.

[159] K. Navaneet, K. Pourahmadi Meibodi, S. Abbasi Koohpayegani, and H. Pirsiavash. “CompGS: Smaller and Faster Gaussian Splatting with Vector Quantization”. *European Conference on Computer Vision*. Springer. 2024, pp. 330–349.

[160] S. Niedermayr, J. Stumpfegger, and R. Westermann. “Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 10349–10358.

[161] J. Sun, H. Jiao, G. Li, Z. Zhang, L. Zhao, and W. Xing. “3DGStream: On-the-Fly Training of 3D Gaussians for Efficient Streaming of Photo-Realistic Free-Viewpoint Videos”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 20675–20685.

[162] Y. Shi, G. Morin, S. Gasparini, and W. T. Ooi. “LapisGS: Layered Progressive 3D Gaussian Splatting for Adaptive Streaming”. *International Conference on 3D Vision 2025*. 2025. URL: https://openreview.net/forum?id=470WxVD1L3.

[163] J. Yan, R. Peng, Z. Wang, L. Tang, J. Yang, J. Liang, J. Wu, and R. Wang. “Instant Gaussian Stream: Fast and Generalizable Streaming of Dynamic Scene Reconstruction via Gaussian Splatting”. *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025, pp. 16520–16531.

[164] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. “2D Gaussian Splatting for Geometrically Accurate Radiance Fields”. *ACM SIGGRAPH 2024 Conference Papers*. SIGGRAPH ’24. Denver, CO, USA: Association for Computing Machinery, 2024. ISBN: 9798400705250. DOI: 10.1145/3641519.3657428. URL: https://doi.org/10.1145/3641519.3657428.

[165] Z. Zhang, B. Huang, H. Jiang, L. Zhou, X. Xiang, and S. Shen. “Quadratic Gaussian Splatting for Efficient and Detailed Surface Reconstruction”. *arXiv preprint arXiv:2411.16392* (2024).

[166] M. Wei, Q. Wu, J. Zheng, H. Rezatofighi, and J. Cai. “Normal-GS: 3D Gaussian Splatting with Normal-Involved Rendering”. *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=kngLs5H6l1.

[167] Y. Jiang, J. Tu, Y. Liu, X. Gao, X. Long, W. Wang, and Y. Ma. “GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 5322–5332.

[168] Z. Liang, Q. Zhang, Y. Feng, Y. Shan, and K. Jia. “GS-IR: 3D Gaussian Splatting for Inverse Rendering”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 21644–21653.

[169] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. “Shape-from-Shading: A Survey”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.8 (1999), pp. 690–706. DOI: 10.1109/34.784284.

[170] R. J. Woodham. “Photometric Method For Determining Surface Orientation From Multiple Images”. *Optical Engineering* 19.1 (1980), p. 191139. DOI: 10.1117/12.7972479. URL: https://doi.org/10.1117/12.7972479.

[171] S. Zhang. "High-Speed 3D Shape Measurement with Structured Light Methods: A Review". *Optics and Lasers in Engineering* 106 (2018), pp. 119–131. ISSN: 0143-8166. DOI: https://doi.org/10.1016/j.optlaseng.2018.02.017. URL: https://www.sciencedirect.com/science/article/pii/S0143816617313246.

[172] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. "FaceWarehouse: A 3D Facial Expression Database for Visual Computing". *IEEE Transactions on Visualization and Computer Graphics* 20.3 (2013), pp. 413–425.

[173] T. Gerig, A. Morel-Forster, C. Blumer, B. Egger, M. Luthi, S. Schönborn, and T. Vetter. "Morphable face models-an open framework". *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018).* IEEE. 2018, pp. 75–82.

[174] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. "Learning a Model of Facial Shape and Expression From 4D Scans". *ACM Trans. Graph* (2017). DOI: 10.1145/3130800.3130813.

[175] B. Egger, S. Sutherland, S. C. Medin, and J. Tenenbaum. "Identity-Expression Ambiguity in 3D Morphable Face Models". *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021).* IEEE. 2021, pp. 1–7.

[176] W. A. Smith, A. Seck, H. Dee, B. Tiddeman, J. B. Tenenbaum, and B. Egger. "A Morphable Face Albedo Model". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 5011–5020.

[177] L. Tran and X. Liu. "Nonlinear 3D Face Morphable Model". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 7346–7355.

[178] L. Tran and X. Liu. "On Learning 3D Face Morphable Model from In-the-Wild Images". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (2021), pp. 157–171. DOI: 10.1109/TPAMI.2019.2927975.

[179] H. Feng, T. Bolkart, J. Tesch, M. J. Black, and V. Abrevaya. "Towards Racially Unbiased Skin Tone Estimation via Scene Disambiguation". *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIII.* Springer. 2022, pp. 72–90.

[180] T. Yenamandra, A. Tewari, F. Bernard, H. Seidel, M. Elgharib, D. Cremers, and C. Theobalt. "i3DMM: Deep Implicit 3D Morphable Model of Human Heads". *Proceedings of the IEEE / CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June 2021.

[181] Y. Zhuang, H. Zhu, X. Sun, and X. Cao. "MoFaNeRF: Morphable Facial Neural Radiance Field". *European Conference on Computer Vision.* 2022.

[182] Y. Xu, L. Wang, Z. Zheng, Z. Su, and Y. Liu. "3D Gaussian Parametric Head Model". *Computer Vision – ECCV 2024.* Ed. by A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol. Cham: Springer Nature Switzerland, 2025, pp. 129–147.

[183] Romdhani and Vetter. "Efficient, Robust and Accurate Fitting of a 3D Morphable Model". *Proceedings Ninth IEEE International Conference on Computer Vision.* 2003, 59–66 vol.1. DOI: 10.1109/ICCV.2003.1238314.

[184] G. Hu, F. Yan, J. Kittler, W. Christmas, C. H. Chan, Z. Feng, and P. Huber. "Efficient 3D Morphable Face Model Fitting". *Pattern Recognition* 67 (2017), pp. 366–379. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2017.02.007.

[185] E. Richardson, M. Sela, R. Or-El, and R. Kimmel. “Learning Detailed Face Reconstruction from a Single Image”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1259–1268.

[186] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Pérez, and C. Theobalt. “MoFA: Model-Based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction”. *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3735–3744. DOI: 10.1109/ICCV.2017.401.

[187] C. Cao, D. Bradley, K. Zhou, and T. Beeler. “Real-Time High-Fidelity Facial Performance Capture”. *ACM Trans. Graph.* 34.4 (July 2015). ISSN: 0730-0301. DOI: 10.1145/2766943. URL: https://doi.org/10.1145/2766943.

[188] A. Chen, Z. Chen, G. Zhang, K. Mitchell, and J. Yu. “Photo-Realistic Facial Details Synthesis from Single Image”. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9429–9439.

[189] P. Dou, S. K. Shah, and I. A. Kakadiaris. “End-to-End 3D Face Reconstruction with Deep Neural Networks”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5908–5917.

[190] R. A. Güler, G. Trigeorgis, E. Antonakos, P. Snape, S. Zafeiriou, and I. Kokkinos. “DenseReg: Fully Convolutional Dense Shape Regression In-the-Wild”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2614–2623. DOI: 10.1109/CVPR.2017.280.

[191] L. Tran, F. Liu, and X. Liu. “Towards High-Fidelity Nonlinear 3D Face Morphable Model”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1126–1135.

[192] B. Gecer, S. Ploumpis, I. Kotsia, and S. Zafeiriou. “GANFIT: Generative Adversarial Network Fitting for High Fidelity 3D Face Reconstruction”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1155–1164.

[193] R. Slossberg, G. Shamai, and R. Kimmel. “High Quality Facial Surface and Texture Synthesis via Generative Adversarial Networks”. *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.

[194] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. “Accelerating 3D Deep Learning with PyTorch3D”. *arXiv:2007.08501* (2020).

[195] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob. “Mitsuba 2: A Retargetable Forward and Inverse Renderer”. *ACM Transactions on Graphics* 38.6 (2019), pp. 1–17.

[196] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. “Modular Primitives for High-Performance Differentiable Rendering”. *ACM Transactions on Graphics (ToG)* 39.6 (2020), pp. 1–14.

[197] T. Deliot, E. Heitz, and L. Belcour. “Transforming a Non-Differentiable Rasterizer into a Differentiable One with Stochastic Gradient Estimation”. *Proc. ACM Comput. Graph. Interact. Tech.* 7.1 (May 2024). DOI: 10.1145/3651298. URL: https://doi.org/10.1145/3651298.

[198] M. Zheng, H. Yang, D. Huang, and L. Chen. “ImFace: A Nonlinear 3D Morphable Face Model with Implicit Neural Representations”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pp. 20343–20352.

[199] M. Li, H. Huang, Y. Zheng, M. Li, N. Sang, and C. Ma. “Implicit Neural Deformation for Sparse-View Face Reconstruction”. *Computer Graphics Forum.* Vol. 41. 7. Wiley Online Library. 2022, pp. 601–610.

[200] E. Ramon, G. Triginer, J. Escur, A. Pumarola, J. Garcia, X. Giro-i-Nieto, and F. Moreno-Noguer. “H3D-Net: Few-Shot High-Fidelity 3D Head Reconstruction”. *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pp. 5620–5629.

[201] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos. “Large Pose 3D Face Reconstruction from a Single Image via Direct Volumetric CNN Regression”. *2017 IEEE International Conference on Computer Vision (ICCV).* 2017, pp. 1031–1039. DOI: 10.1109/ICCV.2017.117.

[202] S. Sharma and V. Kumar. “Voxel-based 3D face reconstruction and its application to face recognition using sequential deep learning”. *Multimedia Tools Appl.* 79.25–26 (July 2020), pp. 17303–17330. ISSN: 1380-7501. DOI: 10.1007/s11042-020-08688-x. URL: https://doi.org/10.1007/s11042-020-08688-x.

[203] M. C. Bühler, K. Sarkar, T. Shah, G. Li, D. Wang, L. Helminger, S. Orts-Escolano, D. Lagun, O. Hilliges, T. Beeler, and A. Meka. “Preface: A Data-Driven Volumetric Prior for Few-Shot Ultra High-Resolution Face Synthesis”. *ICCV.* 2023. DOI: 10.1109/ICCV51070.2023.00315.

[204] J. Luo, J. Liu, and J. Davis. “SplatFace: Gaussian splat face reconstruction leveraging an optimizable surface”. *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).* IEEE. 2025, pp. 774–783.

[205] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial networks”. *Commun. ACM* 63.11 (Oct. 2020), pp. 139–144. ISSN: 0001-0782. DOI: 10.1145/3422622. URL: https://doi.org/10.1145/3422622.

[206] S. An, H. Xu, Y. Shi, G. Song, U. Y. Ogras, and L. Luo. “PanoHead: Geometry-Aware 3D Full-Head Synthesis in 360°”. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2023, pp. 20950–20959. DOI: 10.1109/CVPR52729.2023.02007.

[207] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. “ Zero-1-to-3: Zero-shot One Image to 3D Object ”. *2023 IEEE/CVF International Conference on Computer Vision (ICCV).* Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2023, pp. 9264–9275. DOI: 10.1109/ICCV51070.2023.00853. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.00853.

[208] V. Voleti, C.-H. Yao, M. Boss, A. Letts, D. Pankratz, D. Tochilkin, C. Laforte, R. Rombach, and V. Jampani. “SV3D: Novel Multi-view Synthesis and 3D Generation from a Single Image using Latent Video Diffusion”. *European Conference on Computer Vision.* Springer. 2024, pp. 439–457.

[209] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. “LRM: Large Reconstruction Model for Single Image to 3D”. *The Twelfth International Conference on Learning Representations.* 2024. URL: https://openreview.net/forum?id=sllU8vvsFF.

[210] D. Xie, S. Bi, Z. Shu, K. Zhang, Z. Xu, Y. Zhou, S. Pirk, A. Kaufman, X. Sun, and H. Tan. “LRM-Zero: Training Large Reconstruction Models with Synthesized Data”. *Advances in Neural Information Processing Systems* 37 (2024), pp. 53285–53316.

[211] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia. “Deepfakes and Beyond: A Survey of Face Manipulation and Fake Detection”. *Information Fusion* 64 (2020), pp. 131–148.

[212] X. Wang, K. Wang, and S. Lian. “A survey on face data augmentation for the training of deep neural networks”. *Neural computing and applications* 32.19 (2020), pp. 15503–15531.

[213] T. Li and L. Lin. “AnonymousNet: Natural Face De-Identification With Measurable Privacy”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 56–65. DOI: 10.1109/CVPRW.2019.00013.

[214] S. C. Medin, B. Egger, A. Cherian, Y. Wang, J. B. Tenenbaum, X. Liu, and T. K. Marks. “MOST-GAN: 3D Morphable StyleGAN for Disentangled Face Image Manipulation”. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2. 2022, pp. 1962–1971. DOI: 10.1609/aaai.v36i2.20091.

[215] M. Duff, N. D. Campbell, and M. J. Ehrhardt. “Regularising Inverse Problems with Generative Machine Learning Models”. *Journal of Mathematical Imaging and Vision* 66.1 (2024), pp. 37–56.

[216] F. Mentzer, G. D. Toderici, M. Tschannen, and E. Agustsson. “High-Fidelity Generative Image Compression”. *Advances in Neural Information Processing Systems* (2020).

[217] I. Goodfellow. “NIPS 2016 Tutorial: Generative Adversarial Networks”. *arXiv preprint arXiv:1701.00160* (2016).

[218] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. “Improved Techniques for Training GANs”. *Advances in Neural Information Processing Systems* (2016), pp. 2234–2242.

[219] S. Nowozin, B. Cseke, and R. Tomioka. “f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization”. *Advances in Neural Information Processing Systems* (2016).

[220] M. Arjovsky, S. Chintala, and L. Bottou. “Wasserstein Generative Adversarial Networks”. *International Conference on Machine Learning*. PMLR. 2017, pp. 214–223.

[221] L. Mescheder, A. Geiger, and S. Nowozin. “Which Training Methods for GANs do actually Converge?” *International Conference on Machine Learning*. PMLR. 2018, pp. 3481–3490.

[222] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. *Advances in Neural Information Processing Systems* (2017).

[223] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. “Spectral Normalization for Generative Adversarial Networks” (2018). URL: https://openreview.net/forum?id=B1QRgziT-.

[224] X. Huang and S. Belongie. “Arbitrary Style Transfer in Real-Time With Adaptive Instance Normalization”. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[225] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. “Analyzing and Improving the Image Quality of StyleGAN”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020. DOI: 10.1109/CVPR42600.2020.00813.

[226] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. “Alias-Free Generative Adversarial Networks”. *Proc. NeurIPS.* 2021.

[227] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. “Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling”. *Proceedings of the 30th International Conference on Neural Information Processing Systems.* NIPS’16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 82–90. ISBN: 9781510838819.

[228] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum. “MarrNet: 3D Shape Reconstruction via 2.5D sSetches”. NIPS’17 (2017), pp. 540–550.

[229] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. B. Tenenbaum, and W. T. Freeman. “Visual object networks: image generation with disentangled 3D representation”. *Proceedings of the 32nd International Conference on Neural Information Processing Systems.* NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, pp. 118–129.

[230] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang. “HoloGAN: Unsupervised Learning of 3D Representations from Natural Images”. *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 7588–7597.

[231] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang. “Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images”. *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 2690–2698.

[232] T. Nguyen-Phuoc, C. Richardt, L. Mai, Y.-L. Yang, and N. Mitra. “BlockGAN: Learning 3D Object-Aware Scene Representations from Unlabelled Images”. *Proceedings of the 34th International Conference on Neural Information Processing Systems.* NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[233] S. Lunz, Y. Li, A. Fitzgibbon, and N. Kushman. “Inverse Graphics GAN: Learning to Generate 3D Shapes from Unstructured 2D Data”. *arXiv preprint arXiv:2002.12674* (2020).

[234] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”. *Proceedings of the European Conference on Computer Vision (ECCV).* 2018, pp. 52–67.

[235] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. “A Papier-mâché Approach to Learning 3D Surface Generation”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 216–224.

[236] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia. “Deep Mesh Reconstruction from Single RGB Images via Topology Modification Networks”. *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 9964–9973.

[237] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. “GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis”. *Advances in Neural Information Processing Systems (NeurIPS).* 2020.

[238] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. “Pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis”. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021. DOI: 10.1109/CVPR46437.2021.00574.

[239] Y. Shen and B. Zhou. “Closed-Form Factorization of Latent Semantics in GANs” (2021), pp. 1532–1540. DOI: 10.1109/CVPR46437.2021.00158.

[240] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris. “GANSpace: Discovering Interpretable GAN Controls”. *Proc. NeurIPS*. 2020.

[241] X. Pan, B. Dai, Z. Liu, C. C. Loy, and P. Luo. “Do 2D GANs Know 3D Shape? Unsupervised 3D Shape Reconstruction from 2D Image GANs”. *arXiv preprint arXiv:2011.00844* (2020).

[242] Y. Zhang, W. Chen, H. Ling, J. Gao, Y. Zhang, A. Torralba, and S. Fidler. “Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering”. *International Conference on Learning Representations*. 2021.

[243] S. Sanyal, T. Bolkart, H. Feng, and M. Black. “Learning to Regress 3D Face Shape and Expression from an Image without 3D Supervision”. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[244] Y. Feng, H. Feng, M. J. Black, and T. Bolkart. “Learning an animatable detailed 3D face model from in-the-wild images”. *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–13.

[245] H. Kim, M. Zollhöfer, A. Tewari, J. Thies, C. Richardt, and C. Theobalt. “InverseFaceNet: Deep Monocular Inverse Face Rendering”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4625–4634.

[246] Y. Deng, J. Yang, S. Xu, D. Chen, Y. Jia, and X. Tong. “Accurate 3D Face Reconstruction with Weakly-Supervised Learning: From Single Image to Image Set”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[247] A. Lattas, S. Moschoglou, B. Gecer, S. Ploumpis, V. Triantafyllou, A. Ghosh, and S. Zafeiriou. “AvatarMe: Realistically Renderable 3D Facial Reconstruction" In-the-Wild"”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 760–769.

[248] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. “A 3D Face Model for Pose and Illumination Invariant Face Recognition”. *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*. 2009, pp. 296–301. DOI: 10.1109/AVSS.2009.58.

[249] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li. “Photorealistic facial texture inference using deep neural networks”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5144–5153.

[250] K. Nagano, J. Seo, J. Xing, L. Wei, Z. Li, S. Saito, A. Agarwal, J. Fursund, and H. Li. “paGAN: Real-time Avatars Using Dynamic Textures”. *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–12.

[251] B. Usman, N. Dufour, K. Saenko, and C. Bregler. “PuppetGAN: Cross-Domain Image Manipulation by Demonstration”. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9450–9458.

[252] M. Kowalski, S. J. Garbin, V. Estellers, T. Baltrušaitis, M. Johnson, and J. Shotton. "CONFIG: Controllable Neural Face Image Generation". *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI*. Glasgow, United Kingdom: Springer-Verlag, 2020, pp. 299–315. ISBN: 978-3-030-58620-1. DOI: 10.1007/978-3-030-58621-8_18. URL: https://doi.org/10.1007/978-3-030-58621-8_18.

[253] P. Ghosh, P. S. Gupta, R. Uziel, A. Ranjan, M. J. Black, and T. Bolkart. "GIF: Generative Interpretable Faces" (2020), pp. 868–878. DOI: 10.1109/3DV50981.2020.00097.

[254] M. C. Buehler, A. Meka, G. Li, T. Beeler, and O. Hilliges. "VariTex: Variational Neural Face Textures". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021. DOI: 10.1109/ICCV48922.2021.01363.

[255] J. Piao, K. Sun, K. Lin, and H. Li. *Inverting Generative Adversarial Renderer for Face Reconstruction*. 2021. arXiv: 2105.02431 [cs.CV].

[256] H. Zhou, S. Hadap, K. Sunkavalli, and D. W. Jacobs. "Deep Single-Image Portrait Relighting". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7194–7202.

[257] A. Hou, Z. Zhang, M. Sarkis, N. Bi, Y. Tong, and X. Liu. "Towards High Fidelity Face Relighting with Realistic Shadows". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14719–14728.

[258] R. Abdal, Y. Qin, and P. Wonka. "Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?" *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4432–4441.

[259] Z. Tan, M. Chai, D. Chen, J. Liao, Q. Chu, L. Yuan, S. Tulyakov, and N. Yu. "MichiGAN: Multi-Input-Conditioned Hair Image Generation for Portrait Editing". *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 1–13.

[260] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[261] L. Zhang and D. Samaras. "Face Recognition from a Single Training Image under Arbitrary Unknown Lighting Using Spherical Harmonics". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.3 (2006), pp. 351–363.

[262] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". *ArXiv* (2014). DOI: 10.48550/arXiv.1412.6980.

[263] R. Or-El, S. Sengupta, O. Fried, E. Shechtman, and I. Kemelmacher-Shlizerman. "Lifespan Age Transformation Synthesis". *European Conference on Computer Vision*. Springer. 2020, pp. 739–755.

[264] J. Deng, J. Guo, X. Niannan, and S. Zafeiriou. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". *CVPR*. 2019.

[265] A. Bulat and G. Tzimiropoulos. "How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks)". *International Conference on Computer Vision*. 2017.

[266] Y. Li, C. Huang, and C. C. Loy. “Dense Intrinsic Appearance Flow for Human Pose Transfer”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3693–3702.

[267] L. Hu, C. Ma, L. Luo, and H. Li. “Single-View Hair Modeling Using A Hairstyle Database”. *ACM Transactions on Graphics (ToG)* 34.4 (2015), pp. 1–9.

[268] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241.

[269] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. *arXiv preprint arXiv:1409.1556* (2014).

[270] J. Johnson, A. Alahi, and L. Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. *European Conference on Computer Vision*. Springer. 2016, pp. 694–711.

[271] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[272] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “ImageNet: A large-scale hierarchical image database”. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[273] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. “Image Quality Assessment: From Error Visibility to Structural Similarity”. *IEEE Transactions on Image Processing* (2004). DOI: 10.1109/tip.2003.819861.

[274] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. “The Unreasonable Effectiveness of Deep Features As a Perceptual Metric”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018. DOI: 10.1109/CVPR.2018.00068.

[275] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. “Diffusion Models: A Comprehensive Survey of Methods and Applications”. *ACM Comput. Surv.* 56.4 (Nov. 2023). ISSN: 0360-0300. DOI: 10.1145/3626235. URL: https://doi.org/10.1145/3626235.

[276] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. “Video Diffusion Models”. *Advances in Neural Information Processing Systems* 35 (2022), pp. 8633–8646.

[277] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. “Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets”. *arXiv preprint arXiv:2311.15127* (2023).

[278] T. Yi, J. Fang, J. Wang, G. Wu, L. Xie, X. Zhang, W. Liu, Q. Tian, and X. Wang. “GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 6796–6807.

[279] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. “SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis”. *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=di52zR8xgf.

[280] J. Chen, C. Ge, E. Xie, Y. Wu, L. Yao, X. Ren, Z. Wang, P. Luo, H. Lu, and Z. Li. “Pixart-Σ: Weak-to-Strong Training of Diffusion Transformer for 4K Text-to-Image Generation”. *European Conference on Computer Vision*. Springer. 2024, pp. 74–91.

[281] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. “DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22500–22510.

[282] L. Zhang, A. Rao, and M. Agrawala. “Adding Conditional Control to Text-to-Image Diffusion Models”. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3836–3847.

[283] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. *Advances in Neural Information Processing Systems* (2022), pp. 36479–36494.

[284] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or. “Prompt-to-Prompt Image Editing with Cross-Attention Control”. *ICLR*. 2023. URL: https://openreview.net/forum?id=_CDixzkzeyb.

[285] C. Zhang, C. Zhang, M. Zhang, and I. S. Kweon. “Text-to-Image Diffusion Models in Generative AI: A Survey”. *arXiv preprint arXiv:2303.07909* (2023).

[286] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention Is All You Need”. *Advances in Neural Information Processing Systems* 30 (2017).

[287] F. Xu, Q. Hao, Z. Zong, J. Wang, Y. Zhang, J. Wang, X. Lan, J. Gong, T. Ouyang, F. Meng, et al. “Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models”. *arXiv preprint arXiv:2501.09686* (2025).

[288] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. “Perceiver: General Perception with Iterative Attention”. *International Conference on Machine Learning*. PMLR. 2021, pp. 4651–4664.

[289] J. Zhang, J. Huang, S. Jin, and S. Lu. “Vision-Language Models for Vision Tasks: A Survey”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.8 (2024), pp. 5625–5644. DOI: 10.1109/TPAMI.2024.3369699.

[290] M. Awais, M. Naseer, S. Khan, R. M. Anwer, H. Cholakkal, M. Shah, M.-H. Yang, and F. S. Khan. “Foundational Models Defining a New Era in Vision: A Survey and Outlook”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

[291] H. Ye, J. Zhang, S. Liu, X. Han, and W. Yang. “IP-Adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models”. *arXiv preprint arXiv:2308.06721* (2023).

[292] Y. Han, J. Zhu, K. He, X. Chen, Y. Ge, W. Li, X. Li, J. Zhang, C. Wang, and Y. Liu. “Face Adapter for Pre-Trained Diffusion Models with Fine-Grained ID and Attribute Control”. *European Conference on Computer Vision*. Springer. 2024, pp. 20–36.

[293] X. Peng, J. Zhu, B. Jiang, Y. Tai, D. Luo, J. Zhang, W. Lin, T. Jin, C. Wang, and R. Ji. "PortraitBooth: A Versatile Portrait Model for Fast Identity-preserved Personalization". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 27080–27090.

[294] R. Rekik, S. Wuhrer, L. Hoyet, K. Zibrek, and A.-H. Olivier. "A Survey on Realistic Virtual Human Animations: Definitions, Features and Evaluations". *Computer Graphics Forum*. Vol. 43. 2. Wiley Online Library. 2024, e15064.

[295] L. Simón-Vicente, S. Rodríguez-Cano, V. Delgado-Benito, V. Ausín-Villaverde, and E. C. Delgado. "Cybersickness. A Systematic Literature Review of Adverse Effects Related to Virtual Reality". *Neurología (English Edition)* 39.8 (2024), pp. 701–709.

[296] F. I. Parke and K. Waters. *Computer Facial Animation*. CRC press, 2008.

[297] N. Magnenat-Thalmann and D. Thalmann. "Virtual Humans: Thirty Years of Research, What Next?" *The Visual Computer* 21.12 (2005), pp. 997–1015.

[298] T. Weise, S. Bouaziz, H. Li, and M. Pauly. "Realtime performance-based facial animation". *ACM Trans. Graph.* 30.4 (July 2011). ISSN: 0730-0301. DOI: 10.1145/2010324.1964972. URL: https://doi.org/10.1145/2010324.1964972.

[299] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. A. Beardsley, C. Gotsman, R. W. Sumner, and M. H. Gross. "High-Quality Passive Facial Performance Capture using Anchor Frames". *ACM Trans. Graph.* 30.4 (2011), p. 75.

[300] Y. Zheng, Q. Zhao, G. Yang, W. Yifan, D. Xiang, F. Dubost, D. Lagun, T. Beeler, F. Tombari, L. Guibas, et al. "PhysAvatar: Learning the Physics of Dressed 3D Avatars from Visual Observations". *European Conference on Computer Vision*. Springer. 2024, pp. 262–284.

[301] S. Lombardi, T. Simon, G. Schwartz, M. Zollhoefer, Y. Sheikh, and J. Saragih. "Mixture of Volumetric Primitives for Efficient Neural Rendering". *ACM Trans. Graph* 40.4 (July 2021). ISSN: 0730-0301. DOI: 10.1145/3450626.3459863. URL: https://doi.org/10.1145/3450626.3459863.

[302] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. "Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 5470–5479.

[303] L. Radl, M. Steiner, M. Parger, A. Weinrauch, B. Kerbl, and M. Steinberger. "StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering". *ACM Transactions on Graphics* 4.43 (2024).

[304] O. Alexander, M. Rogers, W. Lambeth, J.-Y. Chiang, W.-C. Ma, C.-C. Wang, and P. Debevec. "The Digital Emily Project: Achieving a Photorealistic Digital Actor". *IEEE Computer Graphics and Applications* 30.4 (2010), pp. 20–31.

[305] J. Kulhanek, M.-J. Rakotosaona, F. Manhardt, C. Tsalicoglou, M. Niemeyer, T. Sattler, S. Peng, and F. Tombari. "LODGE: Level-of-Detail Large-Scale Gaussian Splatting with Efficient Rendering". *arXiv preprint arXiv:2505.23158* (2025).

[306] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann. "MediaPipe: A Framework for Perceiving and Processing Reality". *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*. 2019.

[307] Y. Deng, J. Yang, J. Xiang, and X. Tong. "GRAM: Generative Radiance Manifolds for 3D-Aware Image Generation". *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022. DOI: 10.1145/3592460.

[308] S. C. Medin, G. Li, R. Du, S. Garbin, P. Davidson, G. W. Wornell, T. Beeler, and A. Meka. "FaceFolds: Meshed Radiance Manifolds for Efficient Volumetric Rendering of Dynamic Faces". *Proc. ACM Comput. Graph. Interact. Tech.* 7.1 (May 2024). DOI: 10.1145/3651304. URL: https://doi.org/10.1145/3651304.

[309] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. "D-NeRF: Neural Radiance Fields for Dynamic Scenes". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10318–10327.

[310] L. Song, A. Chen, Z. Li, Z. Chen, L. Chen, J. Yuan, Y. Xu, and A. Geiger. "NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields". *IEEE Transactions on Visualization and Computer Graphics* 29.5 (2023), pp. 2732–2742.

[311] M. Niemeyer and A. Geiger. "GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields". *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[312] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. "Implicit Neural Representations with Periodic Activation Functions". *Advances in Neural Information Processing Systems* 33 (2020).

[313] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. "FiLM: Visual Reasoning with a General Conditioning Layer". *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[314] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. "Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3504–3515. DOI: 10.1109/CVPR42600.2020.00356.

[315] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. "Stereo Magnification: Learning View Synthesis using Multiplane Images". *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201323. URL: https://doi.org/10.1145/3197517.3201323.

[316] C.-h. Wuu et al. "Multiface: A Dataset for Neural Face Rendering". *ArXiv*. 2022. DOI: 10.48550/ARXIV.2207.11243. URL: https://arxiv.org/abs/2207.11243.

[317] Y. Deng, B. Wang, and H.-Y. Shum. "Learning Detailed Radiance Manifolds for High-Fidelity and 3D-Consistent Portrait Synthesis From Monocular Image". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 4423–4433.

[318] H.-B. Duan, M. Wang, J.-C. Shi, X.-C. Chen, and Y.-P. Cao. "BakedAvatar: Baking Neural Fields for Real-Time Head Avatar Synthesis". *ACM Trans. Graph.* 42.6 (Sept. 2023). DOI: 10.1145/3618399. URL: https://doi.org/10.1145/3618399.

[319] M. Atzmon and Y. Lipman. “SAL: Sign Agnostic Learning of Shapes From Raw Data”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2565–2574. DOI: 10.1109/CVPR42600.2020.00264.

[320] R. Du, M. Chuang, W. Chang, H. Hoppe, and A. Varshney. “Montage4D: Interactive Seamless Fusion of Multiview Video Textures”. *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D. ACM, May 2018, pp. 124–133. DOI: 10.1145/3190834.3190843.

[321] M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt. “The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics”. *ACM SIGGRAPH Computer Graphics* 22.4 (1988), pp. 21–30. DOI: 10.1145/378456.378468.

[322] G. E. Blelloch. “Prefix Sums and Their Applications” (1990). DOI: 10.1109/ISTCS.1995.377028.

[323] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. “Pushing the Boundaries of View Extrapolation with Multiplane Images”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 175–184.

[324] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. “Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields”. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023. DOI: 10.1109/ICCV51070.2023.01804.

[325] M. C. Fink, S. A. Robinson, and B. Ertl. “AI-Based Avatars Are Changing the Way We Learn and Teach: Benefits and Challenges”. *Frontiers in Education*. Vol. 9. Frontiers Media SA. 2024, p. 1416307.

[326] Axios. “Morehouse to hire AI teaching assistants”. *Axios Atlanta* (July 2024). URL: https://www.axios.com/local/atlanta/2024/07/04/morehouse-ai-teaching-assistants (visited on 07/29/2025).

[327] Z. Bai, F. Tan, S. Fanello, R. Pandey, M. Dou, S. Liu, P. Tan, and Y. Zhang. “Efficient 3D Implicit Head Avatar with Mesh-anchored Hash Table Blendshapes”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 1975–1984.

[328] Y. Wu, Y. Deng, J. Yang, F. Wei, Q. Chen, and X. Tong. “AniFaceGAN: Animatable 3D-Aware Face Image Generation for Video Avatars”. *NeurIPS* (2022). DOI: 10.48550/arXiv.2210.06465.

[329] S. C. Medin, G. Li, Z. Bai, R. Du, L. Helminger, Y. Zhang, S. Garbin, P. Davidson, G. W. Wornell, T. Beeler, and A. Meka. “LegacyAvatars: Volumetric Face Avatars For Traditional Graphics Pipelines” (2025).

[330] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman. “Volume Rendering of Neural Implicit Surfaces”. *Advances in Neural Information Processing Systems* 34 (2021), pp. 4805–4815.

[331] P.-W. Grassal, M. Prinzler, T. Leistner, C. Rother, M. Nießner, and J. Thies. “Neural Head Avatars From Monocular RGB Videos”. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. DOI: 10.1109/CVPR52688.2022.01810.

[332] Y. Zheng, V. F. Abrevaya, M. C. Bühler, X. Chen, M. J. Black, and O. Hilliges. “I M Avatar: Implicit Morphable Head Avatars From Videos”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022. DOI: 10.1109/CVPR52688.2022.01318.

[333] X. Gao, C. Zhong, J. Xiang, Y. Hong, Y. Guo, and J. Zhang. “Reconstructing Personalized Semantic Facial NeRF Models From Monocular Video”. *ACM Transactions on Graphics (TOG)* 41.6 (2022), pp. 1–12.

[334] S. Ma, Y. Weng, T. Shao, and K. Zhou. “3D Gaussian Blendshapes for Head Avatar Animation”. *ACM SIGGRAPH 2024 Conference Papers*. SIGGRAPH ’24. Denver, CO, USA: Association for Computing Machinery, 2024. ISBN: 9798400705250. DOI: 10.1145/3641519.3657462. URL: https://doi.org/10.1145/3641519.3657462.

[335] S. Athar, Z. Xu, K. Sunkavalli, E. Shechtman, and Z. Shu. “RigNeRF: Fully Controllable Neural 3D Portraits”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022. DOI: 10.1109/CVPR52688.2022.01972.

[336] S. Athar, Z. Shu, and D. Samaras. “FLAME-in-NeRF: Neural Control of Radiance Fields for Free View Face Animation”. *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*. IEEE. 2023, pp. 1–8.

[337] Y. Zheng, W. Yifan, G. Wetzstein, M. J. Black, and O. Hilliges. “PointAvatar: Deformable Point-Based Head Avatars From Videos”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023. DOI: 10.1109/CVPR52729.2023.02017.

[338] C. Wang, D. Kang, Y.-P. Cao, L. Bao, Y. Shan, and S.-H. Zhang. “Neural Point-based Volumetric Avatar: Surface-guided Neural Points for Efficient and Photorealistic Volumetric Head Avatar”. *SIGGRAPH Asia 2023 Conference Papers*. 2023, pp. 1–12.

[339] Y. Xu, B. Chen, Z. Li, H. Zhang, L. Wang, Z. Zheng, and Y. Liu. “Gaussian Head Avatar: Ultra High-Fidelity Head Avatar via Dynamic Gaussians”. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 1931–1941. DOI: 10.1109/CVPR52733.2024.00189.

[340] T. Kirschstein, S. Qian, S. Giebenhain, T. Walter, and M. Nießner. “NeRSemble: Multi-View Radiance Field Reconstruction of Human Heads”. *ACM Trans. Graph* 42.4 (July 2023). ISSN: 0730-0301. DOI: 10.1145/3592455. URL: https://doi.org/10.1145/3592455.

[341] E. Wood, T. Baltrušaitis, C. Hewitt, M. Johnson, J. Shen, N. Milosavljević, D. Wilde, S. Garbin, T. Sharp, I. Stojiljković, et al. “3D Face Reconstruction with Dense Landmarks”. *European Conference on Computer Vision*. Springer. 2022, pp. 160–177.

[342] M. C. Buehler, G. Li, E. Wood, L. Helminger, X. Chen, T. Shah, D. Wang, S. Garbin, S. Orts-Escolano, O. Hilliges, D. Lagun, J. Riviere, P. Gotardo, T. Beeler, A. Meka, and K. Sarkar. “Cafca: High-quality Novel View Synthesis of Expressive Faces from Casual Few-shot Captures”. In: *ACM SIGGRAPH Asia 2024 Conference Paper*. 2024. DOI: 10.1145/3680528.3687580. URL: https://doi.org/10.1145/3680528.

[343] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis. “A hierarchical 3d gaussian representation for real-time rendering of very large datasets”. *ACM Transactions on Graphics (TOG)* 43.4 (2024), pp. 1–15.

[344] T. Kirschstein, J. Romero, A. Sevastopolsky, M. Nießner, and S. Saito. “Avat3r: Large Animatable Gaussian Reconstruction Model for High-fidelity 3D Head Avatars”. *arXiv preprint arXiv:2502.20220* (2025).

[345] Google DeepMind. *Genie 3: A New Frontier for World Models*. https://deepmind.google/discover/blog/genie-3-a-new-frontier-for-world-models/. Accessed: 2025-08-08. Aug. 2025.

[346] Y. Huang and L. F. Yang. “Gemini 2.5 Pro Capable of Winning Gold at IMO 2025”. *arXiv preprint arXiv:2507.15855* (2025).