Towards Photorealistic Dynamic Capture and Animation of Human Hair and Head

Ziyan Wang

CMU-RI-TR-23-77

Sept. 20th, 2023

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Jessica Hodgins, *Chair*
Fernando De La Torre
Jun-Yan Zhu
Michael Zollhoefer, *Meta Reality Labs Research*
Kalyan Sunkavalli, *Adobe Research*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

To my parents Qingjun Hu and Yongfa Chen, and my girlfriend Shi Ouyang

Abstract

Realistic human avatars play a key role in immersive virtual telepresence. A human avatar needs to faithfully reflect human appearance to reach a high level of realism. Besides, a human avatar should also be drivable and express natural motions. Existing works have made significant progress in building drivable realistic face avatars, but they rarely include realistic dynamic hair despite its importance in human appearance. In pursuit of drivable, realistic human avatars with dynamic hair, we focus on the problem of automatically capturing and animating hair from multi-view videos.

We first look into the problem of capturing the motion of the head with near-static hair. Because the hair has complex geometry, we use a *neural volumetric representation* that can be rendered efficiently. As a result, we achieve photorealistic capture of complex hairstyles by optimizing the representation with the gradient from the reconstruction loss on 2D via *differentiable volumetric rendering*.

Then we extend the problem to capturing hair with *dynamics*. To accommodate the complexity introduced by the temporal dimension, *data-priors* on motion like optical flow and point flow are leveraged as additional supervision. To be more specific, we first perform tracking on hair strands with a data prior on motion. In the next step, we attach volumetric primitives to the tracked hair strands to learn the fine-level appearance and geometry via differentiable rendering. We further design a differentiable volumetric rendering algorithm with the optical flow to ensure temporal smoothness at a fine level. As a result, we achieve robust dynamic capture of hair with large motions.

We then address the problem of building a *hair dynamic model* for generating novel animation. We present a two-stage pipeline to build a hair dynamic model in a data-driven manner. The first stage performs hair state compression using an autoencoder-as-a-tracker strategy. The second stage learns a hair dynamic model in a supervised manner using the hair state data from the first stage. The hair dynamic model enables in-the-wild animation of hair that performs hair state transitions conditioned on head motions and head relative gravity direction.

In parallel to capturing and animating specific hairstyles, we explore the problem of how to efficiently capture diverse hair appearances. Hair plays a significant role in personal identity and the efficient creation of personalized avatars with decent hair is essential to individual usages. To handle the large intra-class variance in hair appearance and geometry, we present a universal hair appearance model that focuses on the similarity between different hairstyles in a local region. The model takes 3D-aligned features as input and learns a unified manifold of local hair appearance that adaptively generates appearance for hairstyles with diverse topologies.

Acknowledgements

I want to first express my deep gratitude to my advisor Jessica for her unwavering support, guidance, and mentorship throughout my PhD journey. She played an indispensable role in shaping my PhD research, and her feedback consistently exerted a profound influence on the quality of my dissertation. As my background is originally in computer vision, her expertise in animation and graphics has been instrumental in expanding my horizons. Her profound knowledge and insights have not only enriched my research but have also inspired me to explore new dimensions within my own field. I feel truly fortunate for her great patience with me, and her wisdom enlightened me in both research and life.

I am also grateful to my thesis committee members: Fernando De La Torre, Jun-Yan Zhu, Michael Zollhoefer and Kalyan Sunkavalli, who dedicated their precious time to my thesis and provided many insightful comments. Their groundbreaking research in graphics and computer vision has been both pioneering and influential, offering a wealth of unique perspectives that have profoundly inspired and influenced my own work.

My PhD journey wouldn't be so much interesting without my friends and labmates at CMU. I would like to thank Donglai, Yufei, Gengshan, Chaoyang, Zhengyi, Jiashun, Yi, Xieyang, Yusha, Zhili, Bingbin, Tim, Yuchen, Lingjing, Runliang and Yewen, with whom I had many fruitful discussions in various topics and sparked many intriguing ideas. I would also like to thank my gym buddies Yi, Xieyang, Nan, Zhili, Jinding, Bingda, Shenyuan, Minghan and Kunpeng who worked out with me to boost our mental health and spotted me many times.

Beyond my time at CMU, I feel extremely fortunate and deeply grateful for the opportunity to work at Reality Labs Research Pittsburgh, where I met a lot of great folks. I would like to give my special thanks to my managers Christoph, Michael and Stephen(Steve). They served as great mentors with a solid technical background and a continuous stream of innovative ideas. What is even more precious is their willingness to continuously share their knowledge and experience with junior fellows. They shaped my career significantly and I couldn't imagine what my PhDa journey would be like without them. I would also like to thank my amazing colleagues: Giljoo, Jason, Tuur, Timur, Chen, Tomas, Aljaz, Olivier, He, Yuan, Shoou-I, Gabe, Radu(Alex), Yuefan, Shunsuke, Weipeng, Chenglei and so many more for their help along the path.

Finally, I wish to convey my heartfelt appreciation to my parents, my girlfriend and her family. This thesis is not only a reflection of my work but also a tribute to the unwavering support and love they have provided throughout my academic journey. Their love and unwavering support have consistently acted as a guiding light during my most challenging moments in this academic voyage.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interactive human avatars, representing individuals within a VR/AR environment, introduce the potential for a novel communication paradigm. In pursuit of an immersive experience, human avatars should ideally reflect every nuance of human appearance and motion. Photorealism should facilitate the experience of interacting by making the avatars visually indistinguishable from real people. However, photorealistic avatars can be particularly challenging due to the uncanny valley effect, as they come extremely close to human appearance and subtle differences in their motion can elicit discomfort in users. Achieving a genuine experience of interacting with photorealistic avatars requires them to exhibit natural movements grounded in the principles of physics and human perceptual norms. When using such avatars as digital twins of ourselves, we hope to be able to communicate and interact smoothly with our families and friends regardless of the physical distance, just by putting on a VR headset or sitting in front of a 3D AR display.

Many existing techniques [18, 45, 91, 94] for data-driven avatar creation align face geometric models such as 3DMM [5] or a tracked head mesh with image observations and learn the avatar's appearance by inverse rendering. However, most of those techniques focus on the face and largely ignore the hair. Both the low-quality hair appearance and the lack of motion can make those avatars unrealistic and reduce the sense of presence during interaction. We believe that adding hair with accurate appearance and motion will create a more compelling experience for viewing and engaging with avatars.

In this work, we focus on building an animatable avatar with realistic hair. We argue that hair, as a key part of human appearance, needs to be modeled separately due to the innate uncorrelated variation in both the appearance and motion of hair and head. To achieve controllable and realistic animation of hair, it is important to first understand the geometry, appearance and dynamic properties of the hair. The ability to model those three properties determines the level of realism associated

with a given model. In contrast to the geometry and appearance terms, the dynamics property is not straightforward to capture with a photometric capture system, as it is usually intertwined with the other two. Thus the success in building a controllable and generalizable avatar is closely related to how well we disentangle and reconstruct those three properties: geometry acquisition, appearance modeling and dynamic capture.

One big challenge for capturing realistic hair lies in the the complexity of hair geometry and appearance. The variety of color, curliness and style of human hair is significant. Hair can be tied up or formed into different topologies such as a pony tail, french braid, or a bun. The appearance of hair is quite hard to model due to its complex specular and diffusion properties as light intersects with multiple strands. To model this kind of complex geometry and appearance, we use a volumetric representation. There are several advantages to volumetric representations. First, volumetric representations can model thin and semi-transparent geometry with high fidelity, which is suitable for hair modeling. Second, rendering a volumetric representation is relatively more efficient compared to physics based rendering while still achieving photorealistic rendering quality.

The complexity of hair geometry and appearance is also affected by hair motion. Hair strands are actively driven by the motion of the head, the root of the hair, and gravity. The dynamic state of the hair strands are also affected by interaction with other strands, the scalp that they are rooted on, and the shoulders and the face that they might collide with. These complicated dynamics result in many different hair motions including swinging of clumps of hair, sliding between different layers, and collision with the head. Even with realistic hair geometry, unrealistic physical motion would lead to noticeably unnatural hair for an avatar. Although we found that a volumetric representation is good for modeling the geometry and appearance of near static hair, it is not efficient for modeling motion or temporal alignment at a finer level as it wastes memory on modeling empty spaces. To model complex hair dynamics, we need a hair structure representation that can encapsulate the important hair specific properties.

Beyond reconstructing complex appearance and motion, drivability is desired for animation. To build a avatar with drivable hair, we need a proper driving signal. This signal must be easy to obtain and manipulate. For the face, a general practice is to ask an artist to define a set of key-points and then to use sparse facial key-points as a low dimensional representation for the face and body respectively. Then the complete signal for the face can be decoded from the sparse low dimensional representation using a prior model. With recent advances in visual key-point detection, getting those sparse signals has become easier. However, the innate complexity of hair geometry and the large hair motion state space makes it impractical to manually define a set of sparse key-points that could align different hair states

both semantically and geometrically. Only the root part of a hair strand is attached to the scalp. This structural difference allows the hair to deform more freely than the face, especially for longer hair styles. Although conventional hair animation techniques have achieved good animation quality, the process requires significant manual effort and expert knowledge to create digital hair wigs and select plausible physical parameters for animating them. Furthermore, given the massive number of strands, animating and rendering hair with conventional techniques can be very time consuming.

To address these challenges in building a photorealistic head avatar with dynamic hair, we explored the following directions.

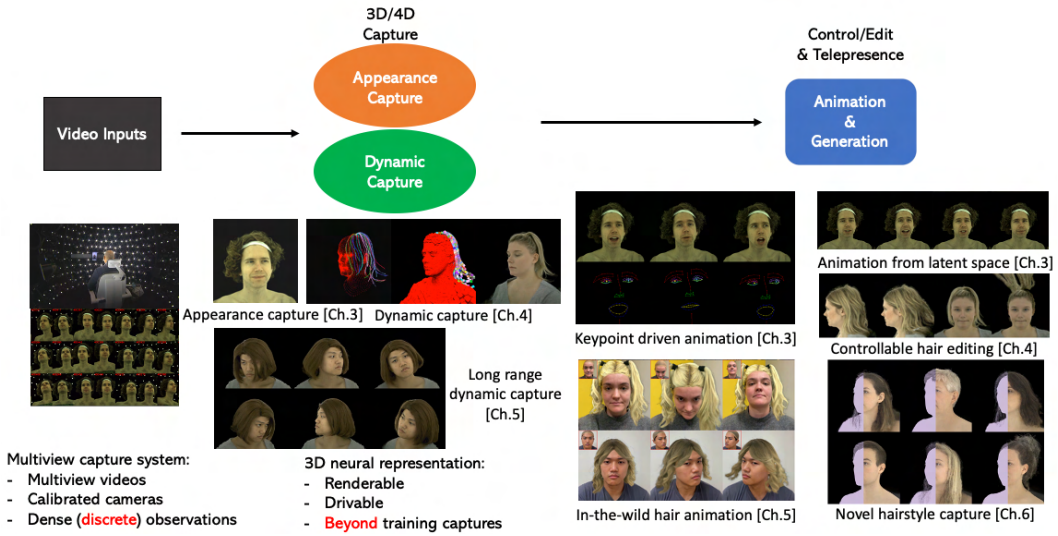

Figure 1.1: **Thesis overview:** Given multiview video captures, we perform appearance and dynamic capture of the upper head with hair. The goal is to learn a 3D neural representation that is both renderable and drivable. Different from the reconstruction, we hope that the 3D neural representation can also help us generate novel content like creating animation and capturing novel hairstyles.

- **Volumetric representation and differentiable rendering.** Volumetric modeling has shown promising results on objects with complex shapes and appearances. In Chapter 3, we explore such good properties of the volume for modeling hair and present a neural volumetric representation for building a photorealistic head avatar with near static hair from a multiview capture system.
- **Hair specific structural information.** Hair exhibits diverse motions which makes capturing dynamic hair a much harder problem than capturing a static

view. In Chapter 4, we study how to combine hair-specific structural information with a neural volumetric representation for dynamic hair capture with better efficiency and accuracy. The explicit modeling of hair strands in the representation makes the hair avatar drivable by sparse signals such as guide hair strands and provides a tangible interface for creating new animation.

- **Data-driven dynamic modeling for animation.** Conventional animation creation requires manual effort to prepare hair geometry and tune physics parameters related to motion. In Chapter 5, we investigate a data-driven alternative to automate the hair animation creation process and create an animatable hair model from the capture of real human hair.

- **A universal prior model for diverse hair appearance capture.** Hairstyle and appearance establish personal identity and are therefore an important part of the personalized avatar. How to efficiently create personalized avatars with diverse hairstyles is critical for modeling individuals in VR/AR applications. In Chapter 6, we explore the possibility of democratizing the creation of personalized avatars with a universal hair appearance model conditioned on sparser inputs. Our model splits different hairstyles into local primitives and builds a prior at that level, which greatly improves the ability to handle a board range of hair topologies over previous techniques.

1.1 Thesis Overview

We now explain each of these works in modeling hair shape, appearance and dynamics. An illustration of the thesis pipeline is shown in Fig. 1.1.

1.1.1 Learning Compositional Radiance Fields of Dynamic Human Heads

In this work, we focus on capturing a dynamic human head with near static hair. There are many studies that apply conventional geometric representations such as meshes and point clouds for avatars with hair. With the advent of neural rendering techniques, neural representations with a backbone of a conventional geometry have been developed to further improve the quality of virtual humans and objects. However, they all have limitations when applied to modeling a head with hair. For example, meshes are good for surfaces but are not efficient for thin and detailed structures such as hair strands. Point clouds can reconstruct the geometry approximately, but they have to be very densely sampled to model dense hair without creating holes.

In pursuit of completeness for hair structures ranging from thin to dense, we explore volumetric models. Some of these existing volumetric methods do not produce results with enough fidelity for driveable human models of hair (Neural Volumes) whereas others have extremely long rendering times (NeRF). We propose a novel 3D representation that combines the best of those two methods to produce both higher resolution and faster results. Our representation bridges the gap between discrete and continuous volumetric representations by combining a coarse 3D-structure-aware grid of animation codes with a continuous learned scene function that maps every position and its corresponding local animation code to a view-dependent emitted radiance and local volume density. Differentiable volume rendering is employed to compute photo-realistic novel views of the human head and upper body as well as to train our novel representation end-to-end using only 2D supervision. As a result, we achieved improved results on novel view synthesis both qualitatively and quantitatively. We use image level metrics such as mean squared error (MSE), peak signal-to-noise ratio (PNSR), and learned perceptual image patch similarity (LPIPS) [119] between the rendered image and ground truth image as a measurement for the quality of the captured 3D dynamic head with near static hair. Compared to previous methods, we reconstruct images under novel views with more details preserved and better visual quality on our dynamic head dataset [109], with less memory than Neural Volumes [46] and a much shorter rendering time than NeRF [61]. In addition, our method is also capable of synthesizing novel unseen expressions based on a global animation code, which is useful for animation and avatar generation.

1.1.2 HVH: Learning a Hybrid Neural Volumetric Representation for Dynamic Hair Performance Capture

In this work, we extend the scope of the problem we study from near static to dynamic hair capture. In dynamic hair capture, the data we need to model is lifted from 3D to 4D, which requires our model to have enough capacity to capture the variance introduced in the temporal dimension. While in the previous work we found that a volumetric representation is suitable for modeling complex and thin hair structures in static shape, we learned that an axis-aligned voxel grid is not suitable for capturing hair in motion. One reason is that it wastes memory on modeling empty space. Another reason is that a volumetric field is not suitable for tracking hair with complex motion and fine level details. Other work on spatio-temporal modeling such as Nerfies [69] and NSFF [40] either learns a volumetric warp field or a volumetric flow field to model the temporal variance. But they are mostly limited to small/simple motions and suffer from excessive rendering time.

To address these limitations, we present a hybrid volumetric hair representation with better hair structure awareness and photorealistic rendering quality. The hybrid representation models the dynamic hair in a coarse-to-fine manner. In the coarse level, we use guide hair strands as a hair structure specific representation to model hair geometry and track hair motion. In the fine level, we attach thousands of primitives to those guide hair strands to model the dense appearance and detailed structure of hair. Each primitive can be rendered efficiently, yet realistically, by building on the latest advances in neural rendering. To reliably drive a hybrid representation, we present a novel way of tracking the guide hair strands using photometric clues. To better improve the temporal consistency and generalization ability of our model at the fine level, we further optimize the 3D scene flow of our representation with multiview optical flow, using volumetric raymarching. As a result, we achieve better rendering qualities on novel views of hair dynamics sequences compared to naive volumetric based methods for spatio-temporal modeling. As our representation can track hair at a sparse strand level, we found that our method 4 can create realistic renders on sequences with large motions such as hair swinging and hair sweeping over shoulders. As a result of enforcing temporal consistency at both a coarse and fine level, the fine level appearance field does not need to change dramatically from frame to frame to accommodate the temporal changes. A by-product of the 3D scene flow optimization is that we achieved a certain level of disentanglement between the hair and non-hair part even without using hair segmentation. In addition to that, the coarse level guide hair strands provide tangible interfaces to create animations with new hair configurations.

1.1.3 NeuWigs: A Neural Dynamic Model for Volumetric Hair Capture and Animation

In this work, we study the problem of hair animation by learning a neural dynamic model for hair. In the graphics community, hair animations have been generated via simulation. The process requires artists to create hair wigs and iteratively tune the parameters required to produce the desired motion. To automate the process, we seek to use data-driven methods and learn a neural dynamic model for hair from real world capture of hair in motion.

A baseline for learning-based hair motion creation is to learn an encoder-decoder model that transfers a sparse driving signal into a dense 3D hair representation in a per-frame manner, without modeling dynamics. This design is straightforward and performs animation and reconstruction well when the per-frame driving signal is well acquired. However, there are several challenges to applying this pipeline. First, the pipeline relies on sophisticated driving signals, such as multi-view im-

ages, a tracked mesh of the hair, or tracked guide hair strands, which are hard to acquire. Furthermore, from an animation perspective, these models are limited to rendering hair based on observations and cannot be used to generate novel motion. For virtual telepresence applications, participants might be wearing a headset that restricts the hair, making it impossible to get the driving signals for the motion being performed. We might also want to animate hair for a bald person or apply a novel hair style where there is no corresponding driving signal. Thus, it is important to develop a dynamic model for hair that does not rely on a conditional driving signal directly coming from real hair.

To address these problems, we present a two-stage approach that models hair motion in a data-driven manner independently from the head motion. The first stage, state compression, learns a low-dimensional latent space of 3D hair states containing motion and appearance, via a novel autoencoder-as-a-tracker strategy. To better disentangle the hair and head in appearance learning, we employ multi-view hair segmentation masks in combination with a differentiable volumetric renderer. The second stage learns a novel hair dynamics model that performs temporal hair transfer based on the discovered latent codes. To enforce higher stability while driving our dynamics model, we employ the 3D point-cloud autoencoder from the compression stage for de-noising of the hair state. Compared to our previous work on dynamic hair capture (Chapter 4), we achieved improved results on novel view synthesis without requiring artists to prepare guide hair geometry. As a result of using hair segmentation as additional supervision, we achieved explicit disentanglement between hair and head, fully addressing the artifacts created when hair and non-hair volumes collide. On the tracking side, our autoencoder-as-a-tracker strategy enjoys better stability over the method in Chapter 4 and supports tracking across discontinuous captures of the same person. On the animation side, the dynamic model is capable of generating new hair animation without direct observation of hair as a driving signal.

1.1.4 A Local Appearance Model for Volumetric Capture of Diverse Hairstyles

In this work, we explore the problem of how to capture novel personalized hairstyles with diverse geometry and appearance. In contrast to the previous chapters where an identity-specific model for dynamic and appearance capture was learned, the focus of this work is to achieve both accurate capture of diverse hairstyles and efficient generation of personalized avatars. There are two major challenges to achieving those goals. The first lies on the capture side, where most conventional multiview capture systems are hard to scale up and not easily accessible to the general public.

We use easy-to-setup sensors which, provide only sparse input information. The second challenge is on the modeling side, where the variance of different hairstyles poses a challenge for accurate capture and generation of personalized avatars with novel hairstyles. To this end, we present a universal hair appearance model for the accurate capture of diverse hairstyles that also serves as a good prior for the efficient generation of novel personalized avatars. The universal hair appearance model generates a personalized avatar with photorealistic hair, conditioned on sparse point clouds with color. To be more specific, the model takes 3D-aligned feature volumes that are diffused from a sparse point cloud and generates a dense radiance field of hair that is spatially aligned with those feature volumes. To better align different hairstyles with diverse topologies and achieve generalization with a limited amount of data, the model learns a hair appearance prior at a local primitive level that is not affected by the overall shape of the hair. Such a design makes the appearance generation independent of global geometry, which makes our model generalizable to diverse hair shapes with similar local hair textures. As a result of better geometric awareness and shape-appearance disentanglement, our methods achieve improved results on capturing a large group of hairstyles jointly, compared to previous state-of-the-art approaches [9, 58]. Extensive experiments also show that our method is capable of generalizing to novel identities and serves as a good initialization for high-quality personalized avatars under sparse view inputs.

1.2 Main Contributions

The contributions of this thesis are as follows:

- We present a novel compositional 3D representation for learning high-quality dynamic neural radiance fields of human heads in motion based on a 3D-structure-aware grid of local animation codes. We develop an importance sampling strategy tailored to human heads that reduces unnecessary computation in free space and enables faster volumetric rendering.
- We present a hybrid neural volumetric representation that binds volumes to guide hair strands for dynamic hair performance capture. We develop a hair tracking algorithm that utilizes multiview optical flow and per-frame hair strand reconstruction while preserving specific geometric properties such as hair strand length and curvature. We implement a volumetric ray marching algorithm on 3D scene flow which enables optimization of the position and orientation of each volumetric primitive through multiview 2D optical flow. We design a hair-specific volumetric decoder for hair volume regression and with awareness of hair structure.

- We present a novel end-to-end data-driven pipeline with a volumetric autoencoder as the backbone for real human hair capture and animation, learned from multi-view RGB images. We learn the hair geometry, tracking and appearance end-to-end with a novel autoencoder-as-a-tracker strategy for hair state compression, where the hair is modeled separately from the head using multi-view hair segmentation. We train an animatable hair dynamic model that is robust to drift using a hair state denoiser realized by the 3D autoencoder from the compression stage.
- We present a novel volumetric feature representation based on a point cloud with color and a local appearance model that is generalizable to various complex hairstyles. We empirically show that our method outperforms previous state-of-the-art approaches in capturing high-fidelity avatars with diverse hairstyles and generating photorealistic appearances for novel identities with challenging hairstyles. The universal model serves as a good prior for getting high-quality personalized avatars under sparse view inputs.

Chapter 2

Related Work

2.1 Neural Geometric Representations

Recently, there have been many works that combine deep neural networks with geometric representations to perform rendering. In this section, we discuss different methods and their trade-offs, categorized by their underlying geometric representation.

Mesh-based Representations: Triangle meshes have been used for decades in computer graphics since they provide an explicit representation of a 2D surface embedded within a 3D space. A primary benefit of this representation is the ability to use high-resolution 2D texture maps to model high-frequency detail on flat surfaces. Recently, differentiable rasterisation [16, 32, 44, 48] has made it possible to jointly optimize mesh vertices and texture using gradient descent based on a 2D photometric re-rendering loss. OpenDR [48] presents an approximately differentiable renderer that is built on an autodifferentiation framework(chumpy). N3MR [32] relaxes the discrete rasterisation process into a differentiable one. SoftRas [44] formulate the aggregation process along the ray in a probabilistic manner, which makes the rasterisation process to be differentiable with respect to occluded objects. DIB-R [16] formulate the rendering as an interpolation between vertex attributes of the closest cover face for efficiency. There are also several work [20, 49, 90–92, 94] that apply differentiable rasterisation to 3D face acquisition from images. MoFA [92] learns to regress a 3DMM [5] face representation from image using differentiable rasterisation. Tran *et al.* [94] learns a non-linear 3DMM model parameteried by deep neural networks using differentiable rasterisation. Genova *et al.* [20] uses additional constraints on identity and cycle consistency to stabilize the training through differentiable rasterization. Tran *et al.* [49] regress both proxy and residual

of face shape and texture with paired regularizations to achieve high fidelity results. Tewari *et al.* [91] learns a per-vertex offsite to the regressed 3DMM geometry and appearance for refinement. FML [90] leverages the implicit multi-view information of the same identity in video for better 3D face retrieval. These methods often require a good initialization of the mesh vertices or strong regularization on the 3D shape to enable convergence. Moreover, these methods require a template mesh with fixed topology which is difficult to acquire. Another drawback is that it is hard to apply mesh and differentiable rasterisation when it comes to model non-surface like geometries like hair.

Point Cloud-based Representations: Point clouds are an explicit geometric representation that lacks connectivity between points, alleviating the requirement of a fixed topology but losing the benefits of 2D texture maps for appearance modeling. Recent works, like [56] and [1], propose methods that generate photo-realistic renderings using an image-to-image translation pipeline that takes as input a deferred shading deep buffer consisting of depth, color, and semantic labels. Similarly, in SynSin [105], per-pixel features from a source image are lifted to 3D to form a point cloud which is later projected to a target view to perform novel view synthesis. Although point clouds are a light-weight and flexible geometric scene representation, rendering novel views using point clouds results in holes due to their inherent sparseness, and it typically requires image-based rendering techniques for in-painting and refinement.

Multi-plane Image-based Representations: Another line of work is using multi-plane images (MPIs) as the scene representation. MPIs [123] are a method to store color and alpha information at a discrete set of depth planes for novel view synthesis, but they only support a restricted range of motion. LLFF [59] seeks to enlarge the range of camera motion by fusing a collection of MPIs [123]. Multi-sphere images (MSIs) [2, 8] are an extension for the use case of stereo 360° imagery in VR, where the camera is located close to the center of a set of concentric spheres.

Voxel-based Representations: One big advantage of voxel-based representations is that they do not require pre-computation of scene geometry and that they are easy to optimize with gradient-based optimization techniques. Many recent works [17, 31, 96, 108] have learned volumetric scene representation based on dense uniform grids. 3D-R2N2 [17] presents a 3D convolutional recurrent network that generates shape in 3D voxel grids conditioned on 2D observations under different camera poses. 3D-GAN [108] learns an auto-encoder on voxels. Tulsiani *et al.* [96] presents a differentiable ray-tracing algorithm for learning to regress voxel with 2D

supervision. LSM [31] learns a multi-view stereo machine with 3D recurrent neural networks, differentiable projection and 2D feature map uplifting. Recently, such volumetric representations have attracted a lot of attention for novel view synthesis. DeepVoxels [84] learns a persistent 3D feature volume for view synthesis with an image-based neural renderer. Neural Volumes [46] proposes a differentiable raymarching algorithm for optimizing a volume, where each voxel contains an RGB and transparency values. The main challenge for voxel-based techniques originates in the cubic memory complexity of the often employed dense uniform voxel grid, which makes it hard to scale these approaches to higher resolutions.

Implicit Geometry Representations: Implicit geometry representations have drawn a lot of attention from the research community due to their low storage requirements and the ability to provide high-quality reconstructions with good generalization. This trend started with geometric reconstruction approaches that first employed learned functions to represent signed distance fields (SDFs) [10, 28, 68] or occupancy fields [21, 55, 71]. DeepSDF [68] and OccNet [55] are among the earliest works that try to learn an implicit function of a scene with an MLP and are fueled by large scale 3D shape datasets, such as ShapeNet [15]. DeepSDF densely samples points around the surface to create direct supervision for learning the continuous SDF, while OccNet learns a continuous occupancy field. ConvOccNet [71] manages to improve OccNet's ability to fit large scale scenes by introducing a convolutional encoder-decoder. ConvOccNet is limited to static scenes and geometry modeling, i.e., it can not handle the dynamic photo-realistic sequences that are addressed by our approach.

Coordinate Based Representations: Inspired by their implicit geometry counterparts, continuous scene representations for modeling colored scenes have been proposed. Scene Representation Networks (SRNs) [85] propose an approach to model colored objects by training a continuous feature function against a set of multi-view images. DVR [64] derived an analytical solution for the depth gradient to learn an occupancy and texture field from RGB images with implicit differentiation. NeRF [61] learns a 5D neural radiance field using differentiable raymarching by computing an integral along each ray. Although promising, their results are limited to a single static scene and the approach is hard to generalize to multiple scenes or a scene with dynamic objects. Another limiting factor is that these representations are extremely costly to render, since every step along the ray requires an expensive evaluation of the complete fully-connected network. As a result of its simplicity and impressive results, many recent works have emerged to improve NeRF in many dimensions like relighting [6, 86, 120], physics-based modeling [117], ren-

dering efficiency [42, 43, 76, 115], generative model [14, 83] and spatio-temporal modeling [39,40,69,70,73,95,99,110,116]. AutoInt [42] speeds up the volumetric raymarching of NeRF by learning closed form solution to the intergral in volumetric raymarching. NSVF [43] and PlenOctrees [115] both optimize the run time efficiency of NeRF by stacking it in a sparse structure like octree. KiloNeRF [76] optimizes the run time efficiency by shortening the model inference time where it substitutes a single MLP by multiple tiny and shallow MLPs. However, the octree structure is optimized specifically for a given static scene, making animation or control of the scene not trivial.

2.2 Hair Modeling

In this section, we discuss the most closely related classical hair dynamic and shape modeling methods.

Image-based Hair Geometry Acquisition is challenging due to the complicated hair geometry, massive number of strands, severe self occlusion and collision and view-dependent appearance. Paris *et al.* [66, 67] and Wei *et al.* [104] reconstruct 3D hair geometry from 2D/3D orientation fields using multi-view images. Luo *et al.* [50, 52] further improve the 3D reconstruction by refining the point cloud from traditional MVS with structure-aware aggregation and strand-based refinement. Luo *et al.* [51] and Hu *et al.* [26] progressively fit hair specific structures like ribbons and wisps to the point cloud. Recently, Nam *et al.* [63] substitute the plane assumption in the conventional MVS by a line-based structure to reconstruct 3D line clouds. Sun *et al.* [87] use OLAT images for more efficient reconstruction of line-based MVS and develop an inverse rendering pipeline for hair that reasons about hair specific reflectance. However, none of those methods explicitly model temporal consistency for a time series capture.

Dynamic Hair Capture. Compared to the vast body of work on hair geometry acquisition, the work on hair dynamics [25, 112, 114, 118] acquisition is much less. Zhang *et al.* [118] uses hair simulation to enforce better temporal consistency over a per-frame hair reconstruction result. However, the simulation parameters are empirically determined and no hair collision is considered. Hu *et al.* [25] solves the physics parameters of a hair dynamics model by running parallel processes under different simulation parameters and adopting the one that best matches the visual observation. However, the computation of this method is relatively heavy. Xu *et al.* [112] performs visual tracking by aligning per-frame reconstruction of hair strands with motion paths of hair strands on a horizontal slice of a video volume. However, this method does not support drivable animation and don't have appearance modeling for hairs. Yang *et al.* [114] developed a deep learning framework

for hair tracking using indirect supervision from 2D hair segmentation and a digital 3D hair dataset. But the results are not photometrically accurate. However those methods mainly focus on geometry modeling and are not photometrically accurate or do not support drivable animation.

Data-driven Hair Animation. Using physics-based simulation for hair animation is a common practice in both academia and film/game industry [4, 103]. However, generating hair animation with physics-based simulation can be costly. To remedy that, reduced data-driven methods [12, 13, 23] simulated only on a small portion of guide hair strands and interpolate the rest of hair strands using skinning weights learned from full simulations. With the advances in deep learning, the efficiency of both dynamic generation [53] and rendering [11, 65] of hair animation has been improved using neural networks. Lyu *et al.* [53] uses deep neural networks for adaptive binding between normal hair and guide hair. Olszewski *et al.* [65] treats hair rendering as an image translation problem and generate realistic rendering on hair conditioned on 2D hair mask and stroke. Similarly, Chai *et al.* [11] achieve faster rendering with photorealistic results by substituting the rendering part in animation pipeline with screen neural rendering techniques. Temporal consistency is enforced in this pipeline by conditioning on hair flow. However, those methods still builds on top of conventional hair simulation pipeline, which requires manual efforts from artist to setup the simulation scene. Wu *et al.* [107] proposed to use secondary motion graph(SDG) for hair animation without relying on conventional hair simulation pipeline during runtime. However, the method is hard to scale up.

2.3 Spatial-temporal Modeling with Coordinate Based Representations

Coordinated based representation have been the major focus of recent literature in 3D learning due to their low memory footprint and ability to dynamically assign the model capacity to the correct regions of 3D space. And there are many extensions on spatio-temporal modeling using coordinate based representations [39, 40, 69, 70, 73, 95, 99, 110]. Non-rigid NeRF [95], D-NeRF [73] and Nerfies [69] introduce a dynamic modeling framework with a canonical radiance field and per-frame warpings. Some works [39, 40, 98, 99, 110, 116] model a 3D video by additionally conditioning the radiance field on temporally varying latent codes or an additional time index. Xian *et al.* [110] further leverages depth as an extra source of supervision. STaR [116] models scenes that consist of a background and one dynamic rigid object. NSFF [40] also combines a static and dynamic NeRF pipeline and uses optical flow to constrain the 3D scene flow derived from the NeRF model of

adjacent time frames. Wang *et al.* [99] introduce a grid of local animation codes for better generalization and improved rendering efficiency. DCT-NeRF [98] learns stable and smooth trajectories of each point in a dynamic scene using NeRF. However, these methods are still limited by either sampling resolution or ability to model complex motions and do not generalize well to unseen motions.

2.4 Volumetric Avatar

There has been a long line of research on learning avatars with volumetric or coordinate based representations. To the best of our knowledge, Neural Volumes [46] is the earliest work that learns a volumetric avatar of upper head from multiview images using differentiable volumetric raymarching. One of the strength of this work is that it could directly optimize a volume grid from multiview images and could still have good quality on semi-transparent objects like hair. One of its followup work [100] combines the volumetric representation and coordinate based representation into a hybrid form for better rendering quality and drivability. Another early work on differentiable volumetric rendering is NeRF [61], where they parameterize radiance field implicitly with MLPs instead of using volume grid. Due to the success of NeRF [61] on modeling 3D scene with good appearance from multiple images, there are many work on building avatar with NeRF [19,22,24,30,35,58,69,70, 75, 121, 122]. Pixel-Aligned Avatars [75] utilizes pixel aligned information as additional inputs for NeRF to extend its drivability and generalization over sequence data. KeypointNeRF [58] improves the generalization of avatars and NeRF's robustness to sparse views by using a new spatial encoding techniques with sparse 3D keypoints. Another type of volumetric avatar is composed of a spatial 3D warp field and a canonical NeRF. Nerfies [69] learns a volumetric deformation field and canonical space NeRF for modeling dynamics related changes. HyperNeRF [70] uplifts the deformation field from 3D eculidean space to a high dimensional hyper semantic space to better model large variations in expressions. NeRFace [19] improves the controllability of NeRF based avatars by using a 3D face morphable model to controll the radiance field defined on faces. IM Avatar [121] improves NeRFace in terms of more complete expression based on implicit skinning fields following FLAME [38]. HeadNeRF [24] learns a parametric head model with illumination using NeRF. PointAvatar [122] learns a point cloud based avatar with temporally conditioned volumetric deformation field for capturing a 3D avatar from video. However, those methods mostly assume hair to be rigidly attached to head without motion. And most of them are still limited by the prohibiting rendering time which is also the limitation of NeRF. In contrast to the line of NeRF based avatars, mixture of volumetric primitives (MVP) [47] builds a volumetric representation that

can generate extremely high-quality realtime renderings that look realistic even on challenging materials, like hair and clothing. The key idea is to model a dynamic head by stacking multiple volumetric primitives only on a tracked head mesh, without wasting memories on empty spaces. Instant codec avatar [9] extends MVP to in-the-wild scenarios and unseen identities. By learning a cross-identity hypernetwork that controls the expression and identity change on volumetric avatars from a large corpus of data, the model can be easily adapted to newly captured identites even from an iphone scan. However, in those work, the hair is not separately modeled and the hair motion is not disentangled from the head, making it hard to generalized to novel hair motion or control the hair motions. Another recent work HVH [102] proposed to model hair and head separately where they use a hybrid model of guide hair stands and volumetric primitives. A hair strand structure-aware tracking algorithm based on per-frame 3D flow optimization is presented to track hair at strand level. To refine the tracking results and enforce temporal smoothness, a fine level volumetric raymarching algorithm on dense 3D volumetric scene flow is presented. However, HVH assumes the guide hair motion is given, in which generating the motion of guide hair itself is already a challenging problem in hair animation.

Chapter 3

Learning Compositional Radiance Fields of Dynamic Human Heads

3.1 Introduction

Modeling, rendering, and animating dynamic human heads at high fidelity, for example for virtual reality remote communication applications, is a highly challenging research problem because of the tremendous complexity of the geometry of the human head and the appearance variations of human skin, hair, teeth, and eyes. Skin exhibits subsurface scattering and shows fine-scale geometric pore-level detail, while the human eyes and teeth are both translucent and reflective at the same time. High fidelity modeling and rendering of human hair is challenging due to its thin geometric structure and light scattering properties. Importantly, the face is not static, but changes dynamically with expression and posture.

Recent work on neural rendering learns either discrete or continuous neural scene representations to achieve viewpoint and animation controllable rendering. Discrete neural scene representations are based on meshes [20, 44, 49, 91, 93], point clouds [1, 56, 105], voxel grids [46, 84], or multi-plane images [59, 123]. However, each of these representations has drawbacks: Meshes, even if dynamically textured [45], struggle to model thin and detailed structures, such as hair. Point clouds, by design, do not provide connectivity information and thus lead to undefined signals in areas of sparse sampling, while making explicit occlusion reasoning challenging. Multi-plane images yield photo-realistic rendering results under constrained camera motion, but produce 'stack of cards'-like artifacts [123] when the camera moves freely. Volumetric representations [46] based on discrete uniform voxel grids are capable of modeling thin structures, e.g., hair, using semi-transparency. While these approaches achieve impressive results, they are hard to scale up due to their innate cubic memory complexity.

To circumvent the cubic memory complexity of these approaches, researchers have proposed continuous volumetric scene representations based on fully-connected networks that map world coordinates to a local feature representation. Scene Representation Networks (SRNs) [85] employ sphere marching to extract the local feature vector for every point on the surface, before mapping to pixel colors. This approach is limited to modeling diffuse objects, making it unsuitable for representing human heads at high fidelity.

Neural radiance fields [61] have shown impressive results for synthesizing novel views of static scenes at impressive accuracy by mapping world coordinates to view-dependent emitted radiance and local volume density. A very recent extension [43] speeds up rendering by applying a static Octree to cull free space. While they have shown first results on a simple synthetic dynamic sequence, it is unclear how to extend the approach to learn and render photo-realistic dynamic sequences of real humans. In addition, it is unclear how to handle expression interpolation and the synthesis of novel unseen motions given the static nature of the Octree acceler-

ation structure.

As discussed, existing work on continuous neural 3D scene representations mainly focuses on static scenes and dynamic scene modeling and editing are not directly achievable under the current frameworks. In this work, we propose a novel compositional 3D scene representation for learning high-quality dynamic neural radiance fields that addresses these challenges. To this end, we bridge the gap between discrete and continuous volumetric representations by combining a coarse 3D-structure-aware grid of animation codes with a continuous learned scene function. We start by extracting a global animation code from a set of input images using a convolutional encoder network. The global code is then mapped to a 3D-structure-aware grid of local animation codes as well as a coarse opacity field. A novel importance sampling approach employs the regressed coarse opacity to speed up rendering. To facilitate generalization across motion and shape/appearance variation, in addition to conditioning the dynamic radiance field on the global animation code, we additionally condition it on a local code which is sampled from the 3D-structure-aware grid of animation codes. The final pixel color is computed by volume rendering. In summary, the main contributions of our work are

- A novel compositional 3D representation for learning high-quality dynamic neural radiance fields of human heads in motion based on a 3D-structure-aware grid of local animation codes.
- An importance sampling strategy tailored to human heads that reduces unnecessary computation in free space and enables faster volumetric rendering.
- State-of-the-art results for synthesizing novel views of dynamic human heads that outperform competing methods in terms of quality.

3.2 Method

In this section, we introduce our novel compositional representation that combines the modeling power of high-capacity voxel-based representations and the ability of continuous scene representations to capture subtle fine-level detail. In Fig. 3.1 we provide an overview of our approach.

The core of the method is a hybrid encoder-decoder architecture, directly supervised with multi-view video sequences. For a given frame, the encoder takes a sparse set of views, and outputs a global animation code, which describes dynamic scene information specific to the frame. The global animation code is used to condition the 3D convolutional decoder, which outputs a coarse 3D structure-aware voxel field. In particular, each voxel stores coarse-level opacity, color and localized

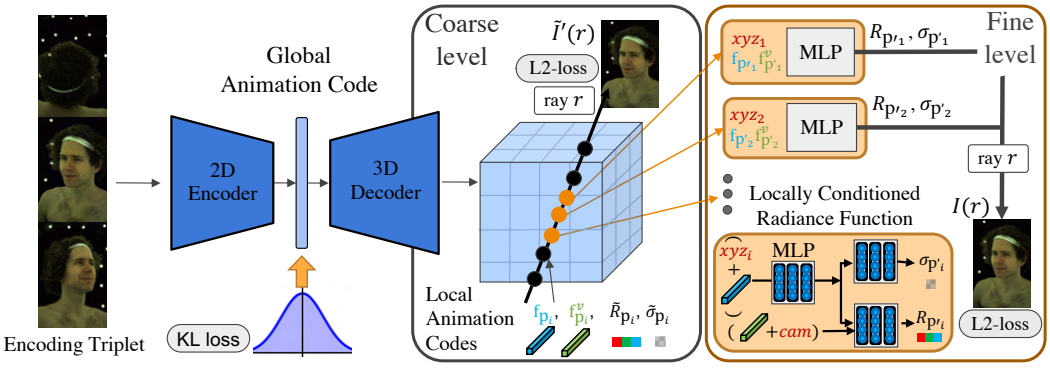

Figure 3.1: **Method overview**. Given a multi-view video as input, we learn a dynamic radiance field parametrized by a global animation code. To render a particular frame, the global code is first mapped to a coarse voxelized field of local animation codes using a 3D convolutional decoder. This grid of animation codes provides local conditioning at each 3D position for the fine-level radiance function, represented as an MLP. Differentiable ray marching is used to render images and provide supervision, and can be sped up significantly by using a ray sampling strategy that uses the coarse grid to determine relevant spatial regions.

animation codes, which represent local dynamical properties of the corresponding spatial region of the scene. The resulting voxel field is further used to create a coarse volumetric rendering of the scene, which may lack fine-level detail, but provides a reliable initial estimate of the scene's geometry, which is crucial to enable efficient continuous scene modeling. To account for the lack of detail, we rely on a continuous scene function, represented as an MLP, to model fine-level radiance. The coarse-level geometry estimate is used to define spatial regions where the function is evaluated, and the local animation codes as spatially-varying conditioning signal to the MLP. To better model view-dependent effects, both coarse- and fine-level representations are partly conditioned on the camera viewpoint. The outputs of the continuous scene function are then used to create the final, refined volumetric rendering of the scene. In what follows, we describe each of the components in detail.

3.2.1 Encoder-Decoder

The goal of the encoder is to produce a compact representation that captures global dynamical properties of the scene, which then serves as a conditioning signal for the decoder. Our encoder is a 2D convolutional network which takes a sparse set of

views and outputs parameters of a diagonal Gaussian distribution $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{256}$. In practice, the encoder is conditioned on three different camera views, concatenated along the channel axis. Given the distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$, we use the reparameterization trick to produce the global animation code $\mathbf{z} \in \mathbb{R}^{256}$ in a differentiable way, and pass it to the decoder. We found that using a variational formulation [34] is critical for making our model animatable.

Given the global animation code $\mathbf{z}$, the goal of the decoder is to produce a coarse-level representation of the scene. In particular, the coarse level is modeled by a volumetric field

$$\mathbf{V}_{\mathbf{p}} = \left(\widetilde{\mathbf{c}}_{\mathbf{p}}, \widetilde{\sigma}_{\mathbf{p}}, \mathbf{f}_{\mathbf{p}}, \mathbf{f}^{v}_{\mathbf{p}}\right) \ , \tag{3.1}$$

where $\widetilde{\mathbf{c}}_{\mathbf{p}} \in \mathbb{R}^3$ is a coarse-level color value, $\widetilde{\sigma}_{\mathbf{p}} \in \mathbb{R}$ is differential opacity, $\mathbf{f}_{\mathbf{p}} \in \mathbb{R}^{32}$ is the view-independent local animation code, $\mathbf{f}^{v}_{\mathbf{p}} \in \mathbb{R}^{32}$ is the view-dependent local animation code, and $\mathbf{p} \in \mathbb{R}^3$ is the spatial location. In our framework, $\mathbf{V}$ is produced by a volumetric decoder as an explicit coarse discrete grid $\mathbf{G} \in \mathbb{R}^{D \times D \times D \times F}$, where $D = 64$ is the spatial dimension of the grid, and $F = 68$ is the dimensionality of the field. Samples $\mathbf{V}_{\mathbf{p}}$ at continuous locations $\mathbf{p} \in \mathbb{R}^3$ are produced with trilinear interpolation over the voxels.

In practice, the decoder is represented by two independent 3D convolutional neural network branches. The first branch is conditioned only on the global code $\mathbf{z}$, and predicts view-independent values, the differential occupancy $\widetilde{\sigma}_{\mathbf{p}}$ and the view-independent local animation codes $\mathbf{f}_{\mathbf{p}}$. The second branch predicts view-dependent color values $\widetilde{\mathbf{c}}_{\mathbf{p}}$ and local animation codes $\mathbf{f}^{v}_{\mathbf{p}}$, and is conditioned on both the global code $\mathbf{z}$ and the viewpoint $\mathbf{v} \in \mathbb{R}^3$, which is computed as a normalized difference between the camera location and the center of the scene.

3.2.2 Volumetric Rendering

Given the discrete voxel field, we apply differentiable ray-marching to obtain coarse volumetric rendering [61]. Namely, for each ray $\mathbf{r} \in \mathbb{R}^3$ shot from the camera center $\mathbf{o} \in \mathbb{R}^3$, we sample N query points $\mathbf{p}_i = (\mathbf{o} + d_i \cdot \mathbf{r})$ along $\mathbf{r}$, where d_i is the depth sampled uniformly between the depth at a near plane d_{min} and a far plane d_{max}. Estimates of expected coarse opacity $\widetilde{A}_{\mathbf{r}}$ and color $\widetilde{I}'_{\mathbf{r}}$ are then computed as

$$\widetilde{A}_{\mathbf{r}} = \sum_{i=1}^{N} T_i \alpha_i \ , \qquad\qquad \widetilde{I}'_{\mathbf{r}} = \sum_{i=1}^{N} T_i \alpha_i \widetilde{\mathbf{c}}_{\mathbf{p}_i} \ , \tag{3.2}$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \widetilde{\sigma}_{\mathbf{p}_j} \delta_j)$, $\alpha_i = (1 - \exp(-\widetilde{\sigma}_{\mathbf{p}_i} \delta_i))$, and $\delta_i = \|d_{i+1} - d_i\|$ is the distance between two neighbouring depth samples. In practice, values $\widetilde{\mathbf{c}}_{\mathbf{p}_i}$, $\widetilde{\sigma}_{\mathbf{p}_i}$ are sampled from the voxel grid with trilinear interpolation.

The final coarse-level rendering is computed by compositing the accumulated color $\widetilde{I}'_{\mathbf{r}}$ and the background color with a weighted sum

$$\widetilde{I}_{\mathbf{r}} = \widetilde{I}'_{\mathbf{r}} + (1 - \widetilde{A}_{\mathbf{r}}) I^{bg}_{\mathbf{r}} \ . \tag{3.3}$$

The resulting coarse rendering roughly captures the appearance of the scene, but lacks fine-level detail. A seemingly straightforward way to improve the level of detail would be to increase the spatial resolution of the voxel grid. Unfortunately, this approach quickly becomes impractical due to the cubic memory complexity of these representations.

3.2.3 Continuous Scene Function

In order to improve fine-level modeling capabilities while avoiding heavy memory costs associated with high-res voxel representations, we introduce a *continuous* scene function $f(\cdot)$, parameterized as an MLP. The key intuition is that voxel-based approaches represent scenes *explicitly* and uniformly across space, thus often wasting resources on irrelevant areas. On the other hand, continuous representations are *implicit*, and allow for more flexibility, as the scene function can be evaluated at arbitrary locations. When combined with a sampling strategy that focuses only on relevant spatial locations, this flexibility can bring significant efficiency improvements.

One crucial difference of our method with respect to the existing continuous neural rendering approaches [61, 83], is that in addition to conditioning on the location, view direction and the global scene information, our scene function is also conditioned on spatially-varying local animation codes. As we demonstrate in our experiments in Sec. 3.3, this increases the effective capacity of our model, and allows our model to capture significantly more detail and better generalize across different motion and shape/appearance variations. We also show that this property is especially important for modeling dynamic scenes, as they require significantly more modeling capacity and the naive MLP-based approaches typically fail.

More formally, the scene function $f(\cdot)$ takes as inputs coordinates of a sampled query point $\mathbf{p}$, view vector $\mathbf{v}$, and the corresponding local animation codes $\mathbf{f}_{\mathbf{p}}, \mathbf{f}^{v}_{\mathbf{p}}$, and produces the fine-level color $\mathbf{c}_{\mathbf{p}} \in \mathbb{R}^3$ and the differential probability of opacity $\sigma_{\mathbf{p}} \in \mathbb{R}$

$$\mathbf{c}_{\mathbf{p}}, \sigma_{\mathbf{p}} = f(\phi(\mathbf{p}), \phi(\mathbf{v}), \mathbf{f}_{\mathbf{p}}, \mathbf{f}^{v}_{\mathbf{p}}) \ .$$

Feature vectors $\mathbf{f}_{\mathbf{p}}, \mathbf{f}^{v}_{\mathbf{p}}$ are obtained from the the coarse voxel grid via trilinear interpolation, and position $\mathbf{p}$ and view $\mathbf{v}$ vectors are passed through a positional encoding $\phi(\cdot)$, in order to better capture high-frequency information [61].

Fine-level rendering $I_{\mathbf{r}}$ and $A_{\mathbf{r}}$ can then be computed by evaluating $f(\cdot)$ at a number of sampled query points along each ray and applying Eq. (3.2)-(3.3). In the

next section, we discuss our novel sampling scheme that significantly speeds up the rendering process.

3.2.4 Efficient Sampling

Using spatially-varying conditioning allows us to increase effective capacity of our continuous scene representation and leads to better generalization. However, producing a high-quality rendering still requires evaluating the scene function at a large number of query locations, which can be computationally expensive [61], and ultimately suffers from similar limitations as the voxel fields. Luckily, we can exploit the fact that our coarse voxel field already contains information about the scene's geometry. To this end, we introduce a simple and efficient sampling scheme, which uses the coarse opacity values to produce a strong initial prior on the underlying geometry. In particular, for each ray $\mathbf{r}$, we first compute a coarse depth $\widetilde{d}_{\mathbf{r}}$ as

$$\widetilde{d}_{\mathbf{r}} = \frac{1}{\widetilde{A}_{\mathbf{r}}} \sum_{i=1}^{N} T_i \alpha_i \cdot d_i \ ,$$

where d_i are the *same* uniform samples as in Eq. (3.2). Then, we obtain our new fine-level location samples from a uniform distribution:

$$d \sim \mathcal{U}\left[\widetilde{d}_{\mathbf{r}} - \Delta_d, \widetilde{d}_{\mathbf{r}} + \Delta_d\right] \ ,$$

centered at the depth estimate $\widetilde{d}_{\mathbf{r}}$, where $\Delta_d = \frac{(d_{max} - d_{min})}{k}$, i.e. $k = 10$ times smaller range than at the coarse level. In Sec. 3.3 we demonstrate that this strategy in practice leads to comparable rendering quality, while being more computationally efficient.

3.2.5 Training Objective

Our model is end-to-end differentiable, which allows us to jointly train our encoder, decoder and the scene MLP, by minimizing the following loss:

$$\mathcal{L} = \mathcal{L}_r + \widetilde{\mathcal{L}}_r + \lambda_f \mathcal{L}_\beta + \lambda_c \widetilde{\mathcal{L}}_\beta + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}} \ .$$

Here $\mathcal{L}_r$ is the error between the rendered and ground truth images for the fine-level rendering:

$$\mathcal{L}_r = \sum_{\mathbf{r} \in \mathcal{R}} ||I_{\mathbf{r}} - I^{gt}_{\mathbf{r}}||^2_2 \ ,$$

where $\mathcal{R}$ is a set of rays sampled in a batch. The coarse-level rendering loss $\widetilde{\mathcal{L}}_r$ is computed similarly. $\mathcal{L}_\beta$ and $\widetilde{\mathcal{L}}_\beta$ are the priors on the fine-level and coarse-level image opacities respectively [46]:

$$\mathcal{L}_\beta = \sum_{\mathbf{r}\in\mathcal{R}} \left(\log A_\mathbf{r} + \log(1 - A_\mathbf{r})\right) \ ,$$

which pushes both the coarse and fine opacities to be sharper, and encodes the prior belief that most of the rays should hit either the object or the background. Finally, the Kullback-Leibler divergence loss $\mathcal{L}_{\mathrm{KL}}$ encourages our global latent space to be smooth [34], which improves the animation and interpolation capabilities of our model.

3.3 Experiments

We first compare with two state-of-the-art methods for novel view synthesis, namely NV [46] and NeRF [61] on four dynamic sequences of a human head making different facial expressions or talking. We then perform an ablation study to test how different feature representations affect the ability to capture longer sequences, as well as the effects of applying different resampling strategies on speed and image quality. We also evaluate generalization capabilities of our model on novel sequence generation and animation, by interpolating in latent space and by driving the model with various input modalities, including keypoints and images.

3.3.1 Datasets

We use a multi-camera system with around 100 synchronized color cameras that produces 2048×1334 resolution images at 30 Hz. The cameras are distributed approximately spherically at a distance of one meter, and focused at the center of the capture system to provide as many viewpoints as possible. Camera intrinsics and extrinsics are calibrated in an offline process. Images are downsampled to 1024×667 for training and testing. Each capture contains $n = 3$ sentences and around $k = 350$ frames in total for each camera view. We trained on $m = 93$ cameras and tested on $q = 33$ frames from another $p = 7$ cameras.

3.3.2 Baselines

We compare our methods with two baselines that we describe in the following.

NV [46]: Neural Volumes performs novel view synthesis of a dynamic object-centric scene by doing raymarching on a warped voxel grid of RGB and differential

opacity that is regressed from three images using an encoder-decoder network. As the volume is conditioned on temporal input of RGB images, NV is capable of rendering dynamic scenes. The volume is of size 128^3 and the warp field is 32^3. The global animation code is a feature vector of 256 entries.

NeRF [61]: NeRF learns a continuous function of scene radiance, including RGB and opacity, with a fully connected neural network conditioned on scene coordinates and viewing direction. Positional encoding is applied to the 3D coordinates to better capture high frequency information, and raymarching is performed to render novel views. Note that the original NeRF approach is not directly applicable to dynamic sequences. Thus, we extend the conditioning signal to NeRF with a global animation code generated from the encoder in NV. The global animation code is generated from the encoder in NV and it is also of size 256.

3.3.3 Training Details

Network Architecture

There are three main neural networks used in our methods: 1) **Encoder**, that regresses image input to the statistics $\boldsymbol{\mu}, \boldsymbol{\sigma}$ of a latent space vector $\mathbf{z} \in \mathbb{R}^{256}$; 2) **Decoder**, a 3D convolutional network that regresses the latent vector $\mathbf{z}$ to a coarse-level volume $\mathbf{V_p}$ of log differential opacity $\widetilde{\sigma}_\mathbf{p}$, color $\widetilde{\mathbf{c}}_\mathbf{p}$, and spatial scene features $\mathbf{f_p}, \mathbf{f}_\mathbf{p}^v$; 3) **Refinement MLP**, that takes in the coordinate of a spatial location $\mathbf{p}$ as well as its corresponding spatial local feature from the coarse-level volume $\mathbf{f_p}, \mathbf{f}_\mathbf{p}^v$ and outputs the fine-level log differential opacity $\sigma_\mathbf{p}$ and color $\mathbf{c_p}$.

For the image encoder and volume decoder, please refer to Table 3.1 and Table 3.2 for their architecture. To better model the view-dependent effects, we employ two decoders to regress the color and opacity at the coarse level. The common structure of each decoder is shown in Table 3.2. For the color decoder, the input is the concatenation of the latent vector $\mathbf{z}\in\mathbb{R}^{256}$ and the camera view direction $\mathbf{v}\in\mathbb{R}^3$, thus the final input size is $N_{in}^c = 256 + 3$ and the output size is $N_{out}^c = 3$, with a parallel branch producing view-dependent spatial scene features $\mathbf{f}_\mathbf{p}^v\in\mathbb{R}^{32}$. Similarly, the opacity decoder only takes the latent vector $\mathbf{z}\in\mathbb{R}^{256}$ as input and regresses opacity $\sigma_p\in\mathbb{R}$ and view-independent spatial scene features $\mathbf{f_p}\in\mathbb{R}^{32}$ from its two branches respectively. To restrict the regressed color $\widetilde{\mathbf{c}}_\mathbf{p}$ to be non-negative, we apply a ReLU function after the last layer that directly outputs it.

In Figure 3.3.3, we show the structure of the Refinement MLP. The spatial scene features $\mathbf{f_p}, \mathbf{f}_\mathbf{p}^v$ are extracted from the feature voxel $\mathbf{V_p}$ with a continuous coordinate $\mathbf{p} \in \mathbb{R}^3$ using tri-linear interpolation. Log differential opacity $\sigma_\mathbf{p} \in \mathbb{R}$ is regressed from the last fully-connected layer of the top branch and no non-linearity is applied. The spatial color value $\mathbf{c_p}$ is the output of the bottom branch and ReLU is applied

afterwards to guarantee the regressed value is non-negative. At the beginning of the refinement network, a concatenation of a positional encoding of position $\mathbf{p}$ and its corresponding view-independent spatial scene feature $\mathbf{f_p}$. The color branch network learns to explain view-dependent effects by having additional inputs in addition to the positional encoding, such as the camera view $\mathbf{v}$ and view-dependent spatial scene feature $\mathbf{f}_\mathbf{p}^v$ at position $\mathbf{p}$. Note that the adapted version of NeRF, which we us as a baseline, shares exactly the same architecture as shown in Figure 3.3.3, except that instead of $\mathbf{f_p}, \mathbf{f}_\mathbf{p}^v$ it uses the global latent vector $\mathbf{z}$ as additional input.

	Encoder	
1	`Conv2d`(9, 32)	
2	`Conv2d`(32, 64)	
3	`Conv2d`(64, 128)	
4	`Conv2d`(128, 128)	
5	`Conv2d`(128, 256)	
6	`Conv2d`(256, 256)	
7	`Conv2d`(256, 256)	
8	`Flatten`()	
9	`Linear`(256x4x2, 512)	
10	`Linear`(512, 256)	`Linear`(512, 256)

Table 3.1: **Encoder architecture**. Each `Conv2d` layer in the encoder has a kernel size of 4, stride of 2 and padding of 1. After each layer, except for the last two parallel fully-connected layers, a Leaky ReLU [54] activation with a negative slope of 0.2 is applied. The last two parallel fully-connected layers produce, respectively, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.

Hyperparameter Settings

We use Adam [33] with a learning rate $1\mathrm{e}{-4}$, and $\beta_1 = 0.9, \beta_2 = 0.999$. All the models are trained for approximately $70 - 100\mathrm{K}$ iterations, each batch containing 64×64 rays. For each ray, we then uniformly sample 128 query locations for the coarse level, and 32 more locations for the fine level using our sampling scheme. We set $\lambda_f = 0.1, \lambda_c = 0.1$ and $\lambda_{KL} = 0.001$ Training on a sequence of 360 frames under 93 camera views with 1024×667 resolution takes approximately 3-4 days on a single NVidia-V100-32GB GPU. All our models are implemented in PyTorch.

3.3.4 Novel View Synthesis

We show quantitative and qualitative results of novel view synthesis on four dynamic sequences of human heads.

	Decoder	
1	`Linear`(N_{in}^X, 1024)	
2	`Reshape`(1024, 1, 1, 1)	
3	`ConvTrans3d`(1024, 512)	`ConvTrans3d`(1024, 512)
4	`ConvTrans3d`(512, 512)	`ConvTrans3d`(512, 512)
5	`ConvTrans3d`(512, 256)	`ConvTrans3d`(512, 256)
6	`ConvTrans3d`(256, 256)	`ConvTrans3d`(256, 256)
7	`ConvTrans3d`(256, 128)	`ConvTrans3d`(256, 128)
8	`ConvTrans3d`(128, N_{out}^X)	`ConvTrans3d`(128, 32)

Table 3.2: **Decoder architecture**. Each layer is followed by a Leaky ReLU [54] activation with a negative slope of 0.2 except for the last two parallel layers. Each `ConvTrans3d` layer has a kernel size of 4, a stride of 2 and a padding of 1. N_{in}^X stands for the input feature size and N_{out}^X is the output size. X here is a placeholder for color or opacity, $X \in \{c, \sigma\}$.

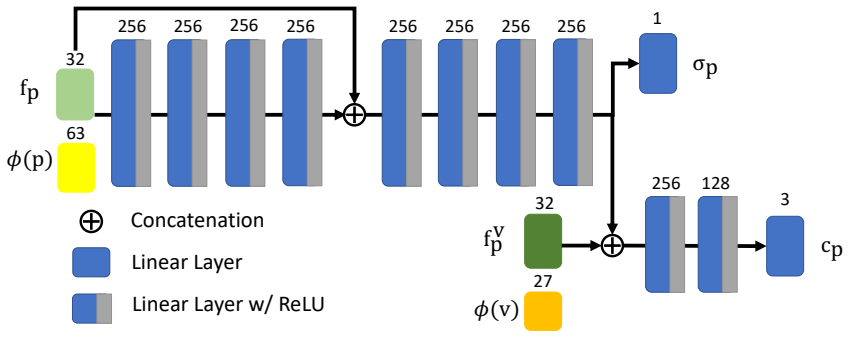

Figure 3.2: **Refinement MLP architecture**. Each blue box is a fully-connected layer and the number on top of each box is the output size of that layer. Blue box with gray tail is a linear layer with ReLU activation. Boxes in other color stand for different inputs and the size is marked on top.

Quantitative Results: We report quantitative evaluation results in Tab. 3.3. Metrics used here are MSE, PSNR, and SSIM. We average those metrics across different test views and time steps, among each of the sequences. To compensate for sensory difference between each camera, we apply the same color calibration network as in NV [46] for our methods as well as all baselines. To compute the parameters of the color calibration networks, we first fit the color calibration model on an additional sentence with all camera views and fix the parameters for all subsequent steps. The first three sequences (Seq1-Seq3) are captures showing the participant

	Sequence1			Sequence2		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM
NV	46.19	31.56	0.8851	52.11	31.24	0.8499
NeRF	43.34	31.88	0.8923	46.89	31.79	0.8531
Ours	**34.01**	**33.09**	**0.9064**	**42.65**	**32.24**	**0.8617**
	Sequence3			Sequence4		
NV	83.07	29.24	0.7742	40.47	32.30	0.9086
NeRF	90.45	28.87	0.7727	35.52	32.95	0.9129
Ours	**79.29**	**29.61**	**0.7826**	**27.62**	**34.12**	**0.9246**

Table 3.3: **Image prediction error**. We compare NV, NeRF, and our method on 4 sequences, and report average error computed over a set of approximately 200 images of 7 views for each sequence. Our method outperforms all other baselines on all metrics.

talking, while the last one (Seq4) is a capture of a range of motions showing challenging expressions. As we can see, our method outperforms all other baselines on the four dynamic sequences in terms of all metrics.

Qualitative Results We show visual comparisons between different models trained on long video sequences in Fig. 3.3 and Fig. 3.4. We can see that NV and NeRF trained on a sequence tend to yield relatively blurry results, while our approach produces sharper images and can reconstruct finer details in terms of both texture and geometry on areas like hair, eyes, and teeth. Our method can achieve better rendering results on video sequence compared to previous methods in terms of photo-realism.

3.3.5 Ablation Studies

Longer Sequences: As one of the major differences between our method and the adapted NeRF is the different feature representation as input for the fine-level neural implicit function, we also tested how this impacts the generalization and fitting power of the approaches. To achieve that, we train our method as well as the temporal conditioned NeRF on sequences with variable length (1, 40, 120, 240, 360 frames) and report their reconstruction performance on a set of views at certain time frames. For all training sets, the first frame is shared and is taken as the test frame. For comparisons, we evaluate three different resolutions (16, 32, 64) for the coarse-level voxel feature in our method to better understand how the voxel resolution could affect the generalization capabilities and expressiveness of our model. Figure 3.6 shows the plot of MSE and SSIM vs. the length of the training sequence

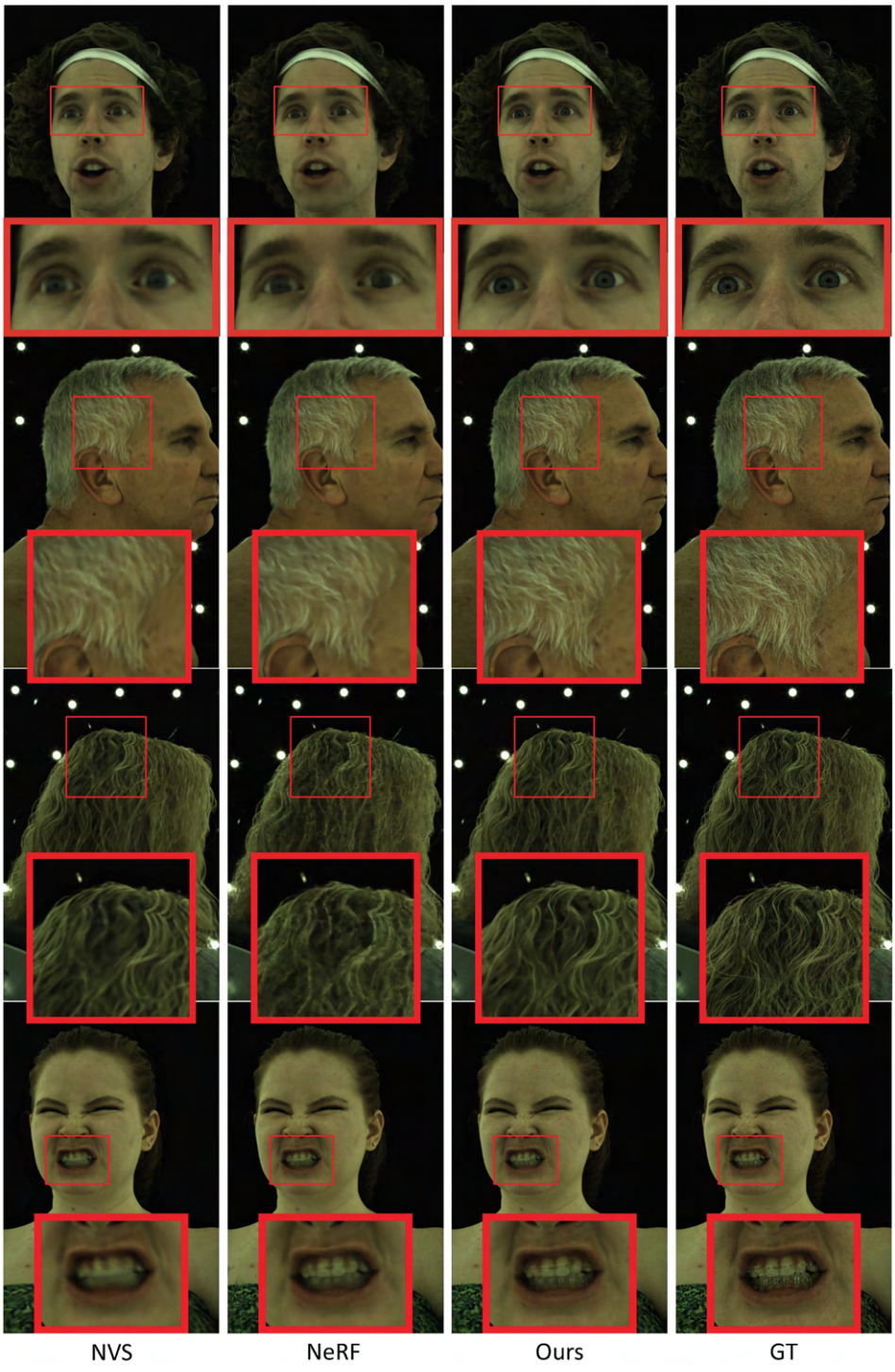

Figure 3.3: **Qualitative comparison of rendered images**. Our method recovers more fine-scale details than NV and NeRF, particularly in high-frequency regions like the eyes and hair. Results are rendered at 1024×667 with insets for better visualization.

	MSE	PSNR	SSIM	Runtime
NeRF+HS	36.33	32.90	0.8898	>25s
NeRF+SS	38.80	32.75	0.8886	19.69s
Ours+HS	**27.23**	**34.24**	0.9090	14.30s
Ours+SS	30.35	34.13	**0.9113**	**3.6s**

Table 3.4: **Ablation on different sampling schemes.** We show image reconstruction results as well as runtime for both NeRF and ours with different sampling strategies.

of different models. A direct visual comparison between models trained on a different number of frames is shown in Figure 3.5. As can be seen, the performance of NeRF with a global animation code drops significantly when the total number of training frames increases, while our method maintains a higher rendering quality due to a more expressive local voxel animation code, which enforces the fine-level implicit function to learn local rather than global representations and capture high frequency details more accurately. In addition, the 3D convolutional decoder imposes 3D inductive bias and improves the capacity of the whole model with the help of a 3D voxel feature tensor that has more spatial awareness compared to a global code. We also see that rendering quality improves and the model achieves better generalization when the coarse-level voxel feature resolution is relatively large. If the resolution is smaller, the performance drops as each local code has to describe a larger region of space, see Fig. 3.5.

Sampling Strategy and Runtime Comparison: We further trained our method and NeRF on a single sentence applying different sampling schemes: the hierarchical sampling (HS) in [61] and our simple sampling (SS). For our method, we use a coarse level voxel resolution of 64^3. Both sampling methods (HS and SS) have 128 points sampled along each ray for coarse level rendering. We show results in Tab. 3.4. As we can see our simple sampling preserves rendering quality while enjoying a large increase in runtime efficiency. For rendering an image with resolution 1024×667, NV takes roughly 0.9s and NeRF is taking >25s whereas our methods takes 3.6s. The improved runtime efficiency stems from the coarse level rendering as our method has to only query the MLP at a small number of positions for refinement. However, our method may fail if the coarse level geometry is too far from the ground truth since we only sample locally. Fine detail, e.g., wrinkles, might not be recovered as geometry, but can still be modeled using view-dependent appearance.

	MSE	PSNR	SSIM
keypoints encoder **w/o** ft	58.52	30.78	0.8891
image encoder **w/o** ft	55.86	31.07	0.8903
keypoints encoder **w/** ft	35.12	32.90	0.9024
image encoder **w/** ft	34.86	33.27	0.9053
full model ft	**32.47**	**33.84**	**0.9121**

Table 3.5: **Novel content synthesis.** We show results on novel content generation and novel sequence fitting. We tested two different encoder models that use data from two modalities: sparse 2D keypoints and images. We use a coarse voxel resolution of 64^3.

3.3.6 Animation

We demonstrate a large variety of applications that is enabled by our approach.

Latent Space Sampling and Interpolation Given the encoder-decoder architecture, we can generate smooth transitions between two expressions by interpolating in latent space and create free-view animations. In Fig. 3.7(b), we show direct interpolation results between two frames with different expressions. The frames in the red and blue bounding box are two key frames and all other frames inbetween are interpolated results. We also show rendering results by randomly sampling in the latent space in Fig. 3.7(c).

We show more results of expression sampling in Figure 3.8 and Figure 3.9. Please also see the videos under "Sampling from Latent Space" and "Interpolation of Sampled Expression" on https://ziyanw1.github.io/hybrid_nerf/. The first video contains 12 uniform keyframe expressions that are directly sampled from the latent space. Then, between each keyframe, we linearly interpolate 10 more frames to create the video. The second videos contains free view rendering of several sampled expressions. As we can see, our model is capable of generating realistic avatars by sampling in the latent space and learnt a continuous latent space with smooth transitions between different latent codes.

Landmark Driven Animation: Because the decoder only depends on a single global code to generate the dynamic field, the original image encoder can be switched to an encoder that takes inputs from other modalities as long as correspondence between input and outputs can be established. To demonstrate controllable animation, we use 2d landmarks as a substitute of the image input and train a simplified PointNet [74]-like encoder that regresses the global code from the set of 2d landmarks. To train such an encoder, we minimize the ℓ_2 distance between the global code

z_{kps} from keypoints and its corresponding global code z_{img} from the image on the training set. Fig. 3.7(a) shows some rendering results that are driven by a keypoint encoder. To test generalization to a novel sentence that is not included in the training data, we deployed the keypoint encoder and the pretrained decoder on a novel sequence. Results on test views are reported in Tab. 3.5. We can see, that with a keypoint encoder using only a regression loss in the latent space, the avatar can be driven with reasonable performance, even though keypoints provide less information than images.

We used a PointNet [74]-like encoder as a base architecture for the keypoint encoder. Compared to the original work, our inputs are different in three aspects: 1) The points are in 2D, 2) The order of each point is fixed rather than arbitrary, 3) All points are roughly aligned to a canonical pose. To simplify the problem, we use the T-Net in the PointNet as the encoder that regresses the latent code from a set of points. We show the architecture in Table 3.6. More results of keypoint-driven animation can be found in Figure 3.10. Please refer to the videos under "Landmark Driven Facial Animation" on https://ziyanw1.github.io/hybrid_nerf/ for more video results. In the video, the 2d keypoints in the blue bounding box are used as input to the keypoint encoder. The image in the middle is the output of our model and the image on the right most column is the ground truth. As we can see, the decoder in our method can also be driven by inputs from other modalities.

	Kps Encoder
1	`Conv1d`(2, 64)
2	`Conv1d`(64, 128)
3	`Conv1d`(128, 256)
4	`Conv1d`(256, 512)
5	`Conv1d`(512, 1024)
6	`MaxPool1d`()
7	`Flatten`()
8	`Linear`(1024, 512)
9	`Linear`(512, 512)
10	`Linear`(512, 256)

Table 3.6: **Keypoint Encoder architecture**. Each layer is followed by a ReLU except for the last fully-connected layer. Each `Conv1d` layer has a kernel size of 1, a stride of 1 and a padding of 0.

Novel Sequence Fitting: To demonstrate our model's ability to generalize to a novel sequence, we show results of animations driven by novel video sequences.

For novel sequence generation from a given input modality, two components need to generalize: (1) the encoder, which produces animation codes given novel image inputs, and (2) the decoder, which renders novel animation codes into images. We first study the generalization ability of the decoder in isolation. To do this, we fine-tune the encoder on the novel sequence, fixing the parameters of the decoder and only back-propagating gradients to the encoder's parameters. Fig. 3.7(d) shows rendering results. To test the ability of generalization to novel input driving sequences, we test the complete encoder-decoder model on a novel sequence, without any fine-tuning. Results are shown in Tab. 3.5. As we can see, an image-based encoder trained with a photometric loss shows better performance on novel content than a key-point encoder trained with a regression loss on the latent space. Innately, image input is a more informative than sparse key-points. Training with a photometric loss rather than a regression loss enables the encoder to output latent codes that are more compatible with the decoder. We also fine-tuned just the image encoder with a photometric loss and we find that the rendering results achieve comparable quality on novel content. We also fine-tuned the full model (encoder and decoder) and we find the gap is small in comparison to the model that only has its encoder fine-tuned.

Please see the videos under "Fitting to a New Sequence" on this page [1]. In both videos, the images in the red bounding boxes serve as inputs. The image in the middle is the output of our model and the image on the right most column is the ground truth. As we can see, the model without finetuning can achieve reasonable performance on fitting the new sequence. And with only encoder finetuning, the encoder quickly adapts to the latent space of the decoder on the novel sequence and creates much smoother results.

3.3.7 Video Results

The video results can be found on our project page 1.

3.4 Limitations

While we achieve state-of-the-art results, our approach is still subject to a few limitations which can be addressed in follow-up work: (1) Our method heavily relies on the quality of the coarse-level voxel field. In cases when the voxel representation has significant errors, the following fine-level model is likely not to recover. (2) Since we rely on the voxel field for our coarse-level representation, our method is

[1] https://ziyanw1.github.io/hybrid_nerf/

primarily applicable to object-centric scenes. Potentially, by substituting the voxelized representation with a coarse depth map, it could also be applied to arbitrary scenes. (3) Although our compositional approach improves the scalability of both voxel-based and continuous representations, our approach is still limited in terms of sampling resolution. Even though the learnt function enables us to go beyond voxel resolution, image quality is still limited by the sampling resolution along the ray. One way to tackle this problem could be to regress the position of a group of voxels, which could serve as a more efficient proxy.

3.5 Conclusion

In this chapter, we proposed a method for rendering and driving photo-realistic avatars of humans captured with a multi-view camera system. Our representation bridges the gap between discrete and continuous volumetric representations by combining a coarse 3D-structure-aware grid of animation codes with a continuous learned scene function that enables high-resolution detail without the need for a dense voxel grid. We show that our approach produces higher-quality results than previous methods, especially as the length of the sequence increases, and is significantly faster than classical neural radiance fields. Our approach also enables driving the model, which we demonstrate via interpolation in the latent space, randomly sampling the latent space, and facial motion control via a set of sparse keypoints. We believe that our approach is a stepping stone towards higher-quality telepresence systems.

Acknowledgments: We thank He Wen and Yuan Dong for helping with key point generation, Zhaoyang Lv and Christoph Lassner for their valuable feedback.

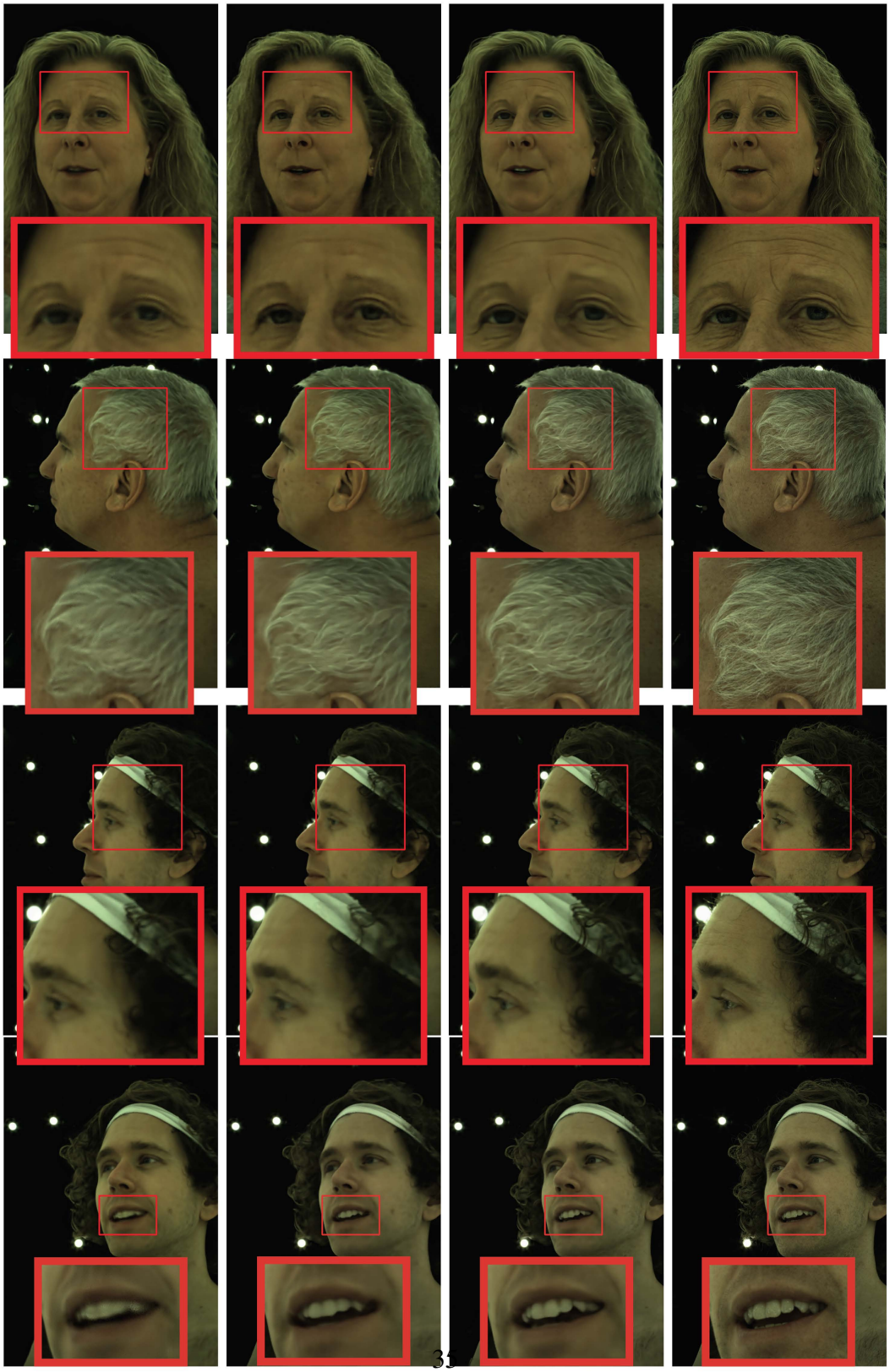

Figure 3.4: **Qualitative comparison of rendered images**. We show more rendering results under novel views. Our method is capable of capturing details like pupil, hair strands and tooth.

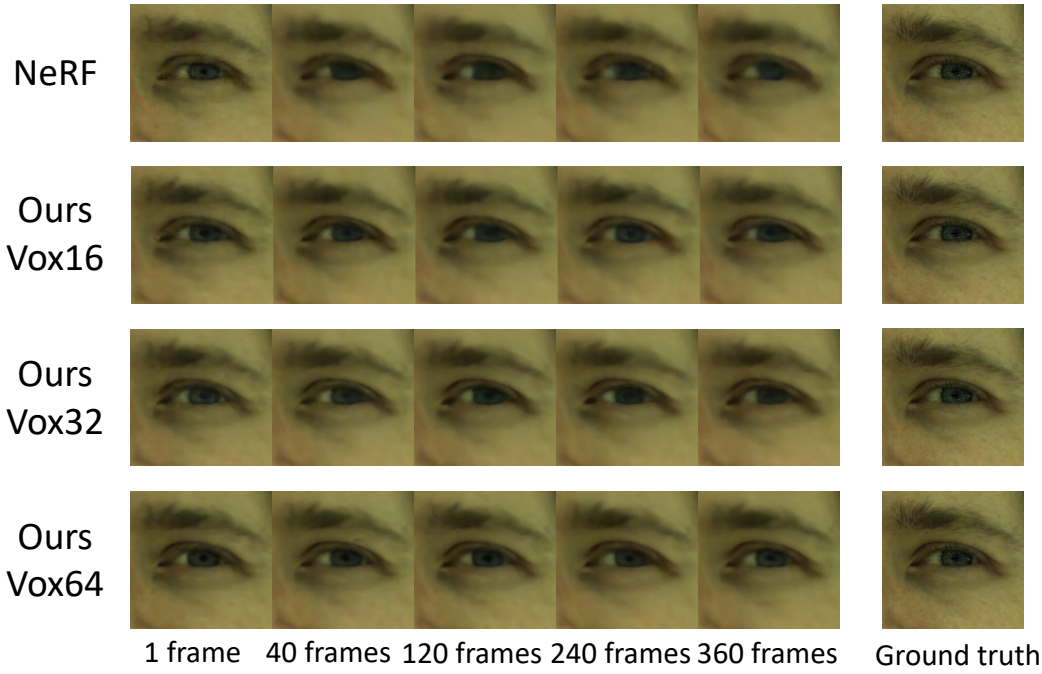

Figure 3.5: **Effect of sequence length on quality.** Conditioning the radiance field on local instead of global animation codes greatly expands model capacity, allowing our model to recover much sharper images even when trained on longer video sequences.

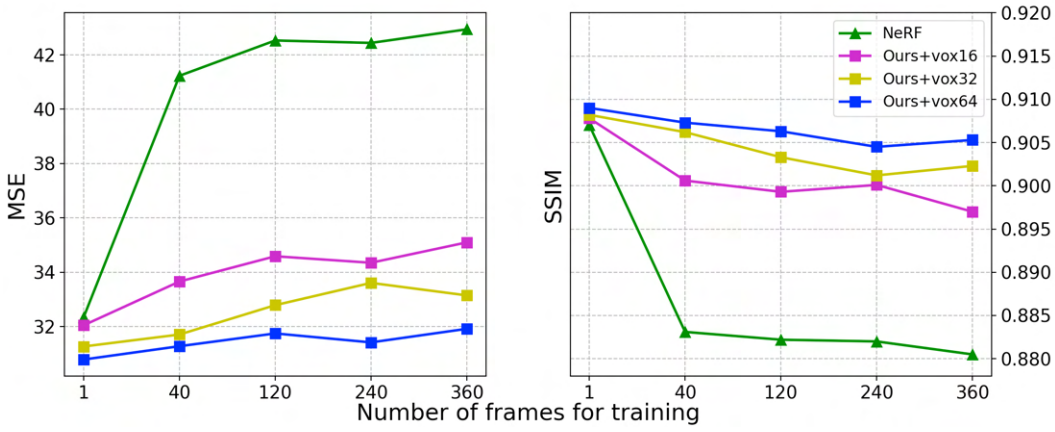

Figure 3.6: **Effect of sequence length on reconstruction.** MSE and SSIM on the first frame v.s. length of the training sequence.

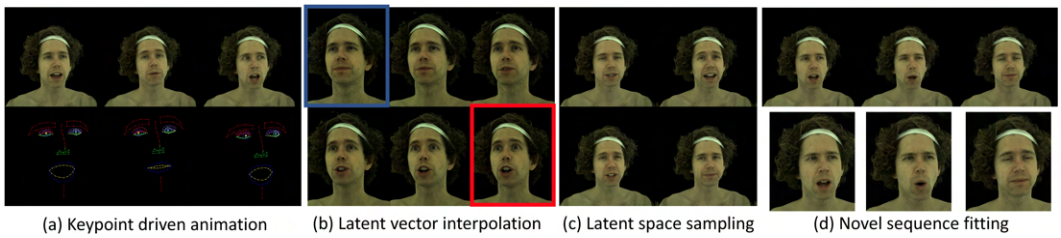

Figure 3.7: **Novel sequence generation**. New animations can be created by dynamically changing the global animation code, for example by (a) using keypoints to drive the animation, (b) interpolating the code at key frames, (c) sampling from the latent distribution, or (d) directly fitting the codes to match a novel sequence.

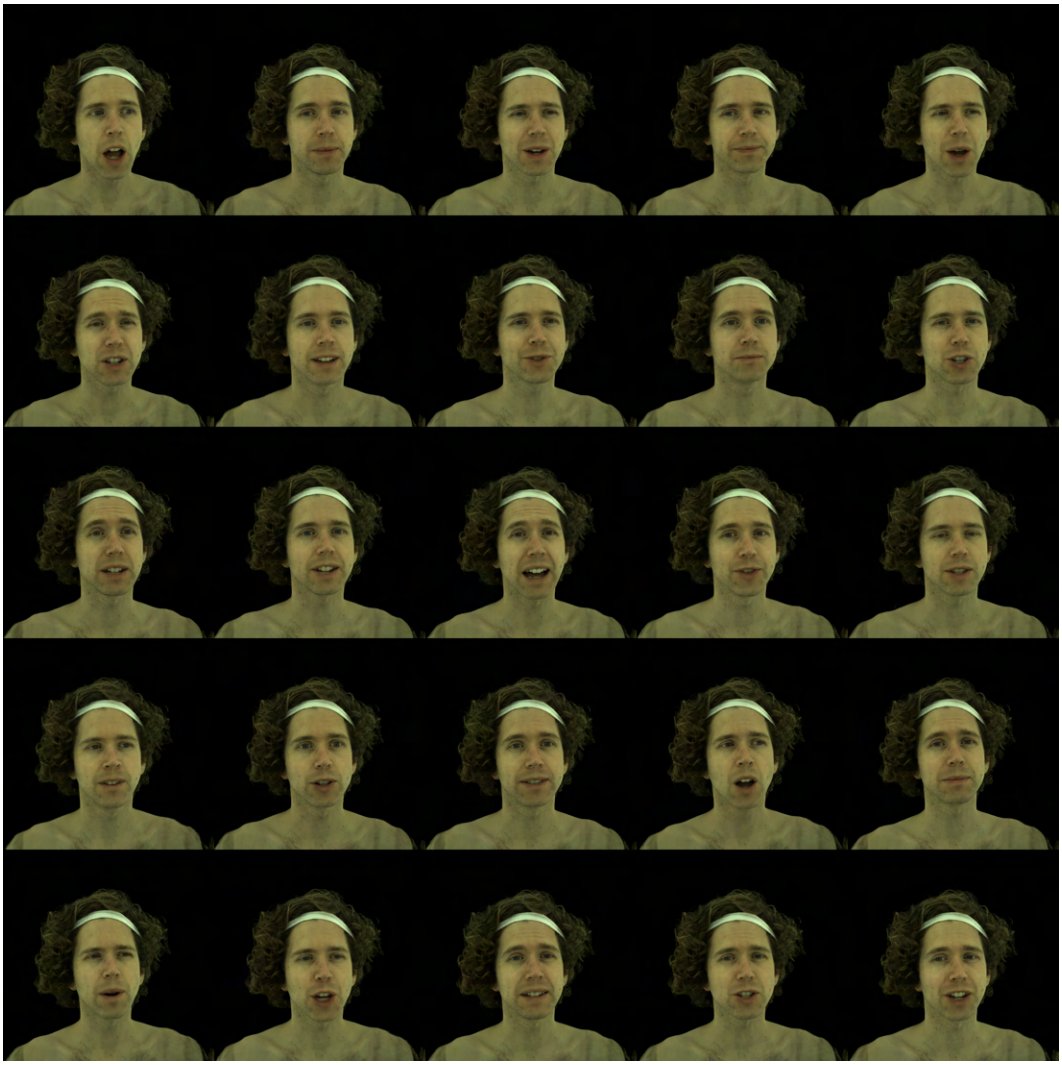

Figure 3.8: **Rendering results of direct sampling in latent space**. We visualize the frontal rendering of the sampled avatars. Each avatar is generated directly from a unique latent code sampled from the latent space.

Figure 3.9: **Rendering results of direct sampling in latent space**. We visualize the sampling results of another identity. Similarly, we sample directly from the latent space and generate the frontal rendering of the avatar corresponding to the sampled code.

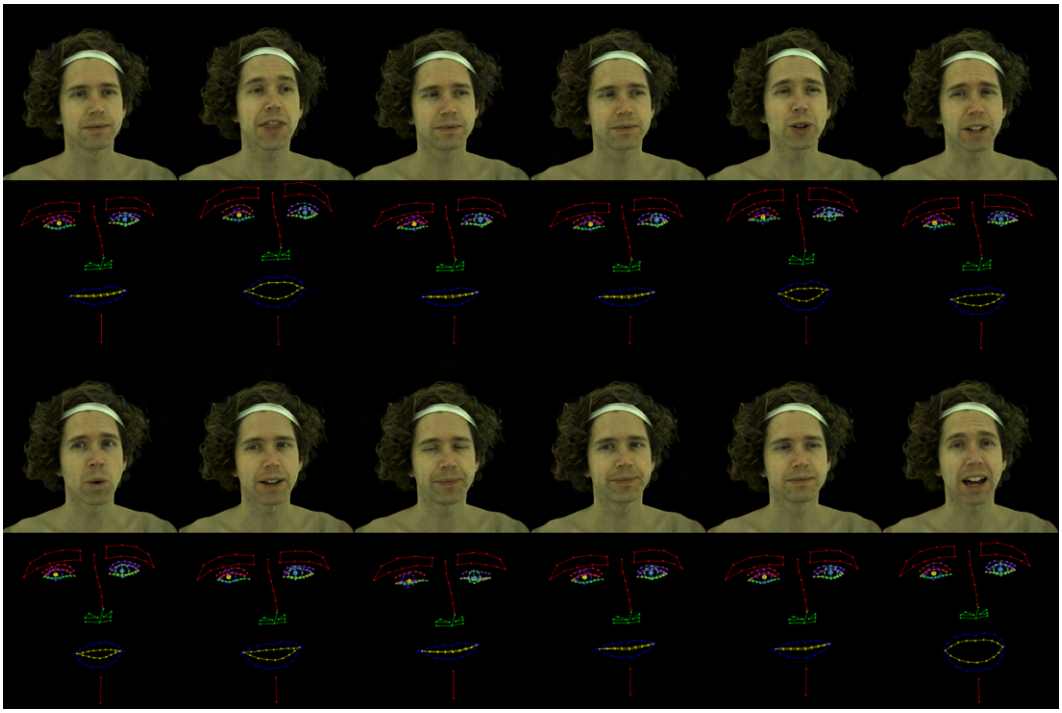

Figure 3.10: **Keypoint-driven animation**. We show rendering results of an avatar driven by facial keypoints. The rendering are shown in the upper row and the driving signal is shown in the lower row. With some fine-tuning on the encoder only, our model can quickly adapt to driving signals from other modality and be reliably driven by facial keypoints.

Chapter 4

HVH: Learning a Hybrid Neural Volumetric Representation for Dynamic Hair Performance Capture

4.1 Introduction

Although notable progress has been made towards the realism of human avatars, cephalic hair is still one of the hardest parts of the human body to capture and render: with usually more than a hundred-thousand components, with complex physical interaction among them and with complex interaction with light, which is extraordinarily hard to model. However, it is an important part of our appearance and identity: hair styles can convey everything from religious beliefs to mood or activity. Hence, hair is critically important to make virtual avatars believable and universally usable.

Previous work on mesh based representations [3, 38, 45, 81, 92, 94, 111] has shown promising results on modeling the face and skin. However, they suffer when modeling hair, because meshes are not well suited for representing hair geometry. Recent volumetric representations [46, 61] have high DoF which allows modeling of a changing geometric structure. They have achieved impressive results in 3D scene acquisition and rendering from multi-view photometric information. Compared to other geometric representations like multi-plane images [2, 8, 60, 88, 123] or point-based representations [1,37,57,78,105], volumetric representations support a larger range of camera motion for view extrapolation and do not suffer from holes when rendering dynamic geometry like point-based representations. Furthermore, they can be learned from multi-view RGB data using differentiable volumetric ray marching, without additional MVS methods.

However, one major flaw of volumetric representations is their cubic memory complexity. This problem is particularly significant for hair, where high resolution is a requirement. NeRF [61] circumvents the $O(n^3)$ memory complexity problem by parameterizing a volumetric radiance field using an MLP. Given the implicit form, the MLP-based implicit function is not limited by spatial resolution. A hierarchical structure with a coarse and fine level radiance function is used and an importance resampling based on the coarse level radiance field is utilized for boosting sample resolution. Although promising empirical results have been shown, they come with at the advance of high rendering time and the quality is still limited by the coarse level sampling resolution. Another limitation of NeRFs is that they were initially designed for static scenes. There is some recent work [39, 40, 69, 70, 73, 95, 99, 110, 116] that extends the original NeRF concept to modeling dynamic scenes. However, they are still limited to relatively small motions, do not support drivable animation or are not efficient for rendering.

We present a hybrid representation: by using many volumetric primitives, we focus the resolution of the model onto the relevant regions of the 3D space. For each of the volumes, we construct a neural representation that captures the local appearance of the hair in great detail, similar to [43,47,76,99] . However, without ex-

plicitly modeling the dynamics and structure of hair, it would be hard for the model to learn these properties solely through the indirect supervision of the multi-view appearance. Given that the model learns to position primitives in an unsupervised manner, the model is also prone to overfitting as a result of not incorporating any temporal consistency during training. We address the problem of spatio-temporal modeling of dynamic upper head and hair by explicitly modeling hair dynamics at the coarse level and by enforcing temporal consistency of the model by multi-view optical flow at the fine level.

Procedurally, we first perform hair strand tracking at a coarse level by lifting multi-view optical flow to a 3D scene flow. To constrain the hair geometry and reduce the impact of the noise in multi-view optical flow, we also make sure the tracked hair strands preserve geometric properties like shape, length and curvature across time. As a second step, we attach volumes to hair strands to model the dynamic scene which can be optimized using differentiable volumetric raymarching. The volumes that are attached to the hair strands are regressed using a decoder that takes per-hair-strand features and a global latent code as input and is aware of the hair specific structure. Additionally, we further enforce fine 3D flow consistency by rendering the 3D scene flow of our model into 2D and compare it with the corresponding ground truth optical flow. This step is essential for making the model generalize better to unseen motions. To summarize, the contributions of this work are

- A hybrid neural volumetric representation that binds volumes to guide hair strands for hair performance capture.
- A hair tracking algorithm that utilizes multiview optical flow and per-frame hair strand reconstruction while preserving specific geometric properties like hair strand length and curvature.
- A volumetric ray marching algorithm on 3D scene flow which enables optimization of the position and orientation of each volumetric primitive through multiview 2D optical flow.
- A hair specific volumetric decoder for hair volume regression and with awareness of hair structure.

4.2 Method

In this section, we introduce our hybrid neural volumetric representation for hair performance capture. Our representation combines both, the drivability of guide

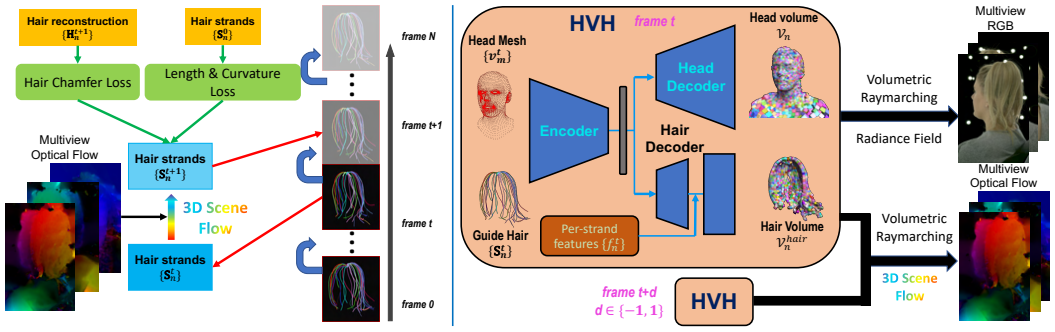

Figure 4.1: **Pipeline.** Our method consists of two stages: in the first stage, we perform guide hair tracking with multiview optical flow as well as per-frame hair reconstruction. In the second stage, we further amplify the sparse guide hair strands by attaching volumetric neural rendering primitives and optimizing them by using the multiview RGB and optical flow data.

hair strands and the completeness of volumetric primitives. Additionally, the guide hair strands serve as an efficient coarse level geometry for volumetric primitives to attach to, avoiding unnecessary computational expense on empty space. As a result of guide hair strand tracking as well as dense 3D scene flow refinement, our model is temporally consistent with better generalization over unseen motions. As illustrated in Fig. 4.1, the whole pipeline contains two major steps which we will explain separately. In the first step, we perform strand-level tracking that leverages multi-view optical flow information and propagates information about a subset of tracked hair strands into future frames. To save computation time, we track only guide hairs instead of tracking all hair strands. This is a widely used technique in hair animation and simulation [13, 27, 72], which leads to a significant boost in run time performance. However, getting the guide hairs tracked is not enough to model the hair motion and appearance or to animate all the hairs due to the sparseness of the guide hairs. To circumvent this, we combine it with a volumetric representation by attaching volumetric primitives to the nodes on the guide hairs. This hybrid representation has good localization of hairs in an explicit way and has full coverage of all the hairs, making use of the benefits of both representations. Another advantage is that the introduction of volumes allows optimizing hair shape and appearance by multi-view dense photometric information via differentiable volumetric ray marching. In the second step, we use the attached volumetric primitives to model the hairs that are surrounding the guide hair strands to achieve dense hair appearance, shape and motion acquisition. A hair specific volume decoder is designed for regressing those volumes, conditioning on both a global latent vector and hair strand feature vectors with hair structure awareness. Additionally, we develop a volumetric ray-

marching algorithm for 3D scene flow that facilitates the learning from multi-view 2D optical flow. We show in the experiments that the introduction of additional optical flow supervision yields better temporal consistency and generalization of the model.

4.2.1 Guide Hair Tracking

We frame the guide hair tracking process as an optimization problem. Given the guide hair strands and multi-view optical flow at the current frame t, we unproject and fuse optical flow under different camera poses into 3D flow and use that to infer the next possible position of the guide hairs at the next frame $t+1$. The guide hair initialization at first frame is prepared by artist.

Data Setup and Notation. In our setting, we perform hair tracking using multi-view video data. We use a multi-camera system with around 100 synchronized color cameras that produces 2048×1334 resolution images at 30 Hz. The cameras are focused at the center of the capture system and distributed spherically at a distance of one meter to provide as many viewpoints as possible. Camera intrinsics and extrinsics are calibrated in an offline process. We generate multi-view optical flow between adjacent frames for each camera, using the OpenCV [7] implementation of [36]. We acquire per-frame hair geometry by running [63]. We parameterize guide hairs as connected point clouds. Given a specific hair strand $\mathbf{S}^t$ at time frame t, we denote the Euclidean coordinate of the $\mathbf{n}$th node on hair strand $\mathbf{S}^t$ as $\mathbf{S}^t_n$. Similarly, we have the future position of $\mathbf{S}^t_n$ at time frame $t+1$ as $\mathbf{S}^{t+1}_n$. Next we introduce the notations for multi-view camera related information. We denote $\Pi_i(\cdot)$ as the camera transformation matrix of camera i which projects a 3D point into 2D image coordinate. We denote $\mathbf{I}_{of,i}$ and $\mathbf{I}_{d,i}$ as 2D matrix of optical flow and depth of camera i respectively. We denote $\mathbf{H}^t_n$ as the reconstructed point cloud with direction from [63, 87]. Unless otherwise stated, all bold lower case symbols denote vectors.

Tracking Objectives. Given camera i, we could project a 3D point into 2D to retrieve its 2D image index. The camera projection is defined as

$$\hat{\mathbf{p}}^t_{s,i} = \begin{bmatrix} \mathbf{p}^t_{s,i} \\ \mathbf{1} \end{bmatrix} = \Pi_i(\mathbf{S}^t_n),$$

where $\hat{\mathbf{p}}^t_{s,i}$ is the homogeneous coordinate of $\mathbf{p}^t_{s,i}$. Given the camera projection formulation, we formulate the first data-term objective based on optical flow as follows:

$$
\begin{aligned}
\mathcal{L}_{of} &= \sum_{n,i} \omega_{n,i} ||\mathbf{S}_n^{t+1} - \mathbf{Z}_i(\mathbf{S}_n^{t+1}) \Pi_i^{-1}(\mathbf{p}_{s,i}^t + \delta_{\mathbf{p}})||_2^2, \\
\omega_{n,i} &= exp(-\sigma ||\mathbf{Z}_i(\mathbf{S}_n^t) - \mathbf{I}_{d,i}(\mathbf{p}_{s,i}^t)||_2^2), \\
\delta_{\mathbf{p}} &= \mathbf{I}_{of,i}(\mathbf{p}_{s,i}^t),
\end{aligned}
$$

where we denote $\mathbf{Z}_i(\cdot)$ as the function that represents the depth of a certain point under camera i and ω_i serves as a weighting factor for view selection where a smaller value means larger mismatch of projected depth and real depth under the ith camera pose. We use a $\sigma = 0.01$.

In parallel with the data-term objective on optical flow, we add another data-term objective to facilitate geometry preserved tracking, which compares the Chamfer distance between tracked guide hair strands and the per-frame hair reconstruction from [63]. This loss is designed to make sure that the guide hair geometry point cloud will not deviate too much from the true hair geometry. Unlike the conventional Chamfer loss, we also penalize the cosine distance between the directions of $\mathbf{S}_n^t$ and the direction of its closest $k = 10$ neighbors as $\mathcal{H}(\mathbf{S}_n^{t+1}) \subsetneq \{\mathbf{H}_n^{t+1}\}$; the losses are defined as:

$$
\begin{aligned}
\mathcal{L}_{hdir} &= \sum_{n,\mathbf{h} \in \mathcal{H}(\mathbf{S}_n^{t+1})} \omega_{n,\mathbf{h}}^d (1 - |\cos(\mathbf{dir}(\mathbf{S}_n^{t+1}), \mathbf{dir}(\mathbf{h}))|), \\
\mathcal{L}_{hpos} &= \sum_{n,\mathbf{h} \in \mathcal{H}(\mathbf{S}_n^{t+1})} \omega_{n,\mathbf{h}}^r ||\mathbf{S}_n^{t+1} - \mathbf{h}||_2^2,
\end{aligned}
$$

where $\omega_{n,\mathbf{h}}^d = exp(-\sigma ||\mathbf{S}_n^{t+1} - \mathbf{h}||_2^2)$ is a spatial weighting, $cos(\cdot, \cdot)$ is a cosine distance function between two vectors and $\mathbf{dir}(\mathbf{S}_n^{t+1}) = \mathbf{S}_{n+1}^{t+1} - \mathbf{S}_n^{t+1}$ is a first order approximation of the hair direction at $\mathbf{S}_n^{t+1}$. $\omega_{n,\mathbf{h}}^r = cos(\mathbf{dir}(\mathbf{S}_n^{t+1}), \mathbf{dir}(\mathbf{h}))$ is a weighting factor that aims at describing the direction similarity between $\mathbf{S}_n^{t+1}$ and $\mathbf{h}$. With $\mathcal{L}_{hdir}$, we could groom the guide hairs $\mathbf{S}_n^{t+1}$ to have similar direction to its closest $k = 10$ neighbors in $\mathcal{H}(\mathbf{S}_n^{t+1})$, resulting in a more consistent guide hair direction distribution. Alternatively, $\mathcal{L}_{hpos}$ guarantees that the tracked guide hairs do not deviate too much from the reconstructed hair shapes.

However, with just the data-term loss, the tracked guide hairs might overfit to noise in the data terms. To prevent this, we further introduce several model-term objectives for hair shape regularization.

$$
\begin{aligned}
\mathcal{L}_{len} =& \sum_n (||\mathbf{dir}(\mathbf{S}_n^{t+1})||_2 - ||\mathbf{dir}(\mathbf{S}_n^0)||_2)^2, \\
\mathcal{L}_{tang} =& \sum_n ((\mathbf{S}_{n+1}^{t+1} - \mathbf{S}_n^{t+1} - \mathbf{S}_{n+1}^t + \mathbf{S}_n^t) \cdot \mathbf{dir}(S_n^t))^2 + \\
& ((\mathbf{S}_{n+1}^t - \mathbf{S}_n^t - \mathbf{S}_{n+1}^{t+1} + \mathbf{S}_n^{t+1}) \cdot \mathbf{dir}(S_n^{t+1}))^2, \\
\mathcal{L}_{cur} =& \sum_n (\mathbf{cur}(\mathbf{S}_n^{t+1}) - \mathbf{cur}(\mathbf{S}_n^0)),
\end{aligned}
$$

where $\mathbf{cur}(\mathbf{S}_n^t)$ is a numerical approximation of curvature at point $\mathbf{S}_n^t$ and is defined as:

$$
\sqrt{\frac{24(||\mathbf{dir}(\mathbf{S}_n^t)||_2 + ||\mathbf{dir}(\mathbf{S}_n^t)||_2 - ||\mathbf{S}_n^t - \mathbf{S}_{n+2}^t||_2)}{||\mathbf{S}_n^t - \mathbf{S}_{n+2}^t||_2^3}}.
$$

We optimize all loss terms together to solve $\{\mathbf{S}_n^{t+1}\}$ given $\{\mathbf{S}_n^t\}$ with:

$$
\begin{aligned}
\mathcal{L}_{hair} =& \mathcal{L}_{of} + \omega_{hdir}\mathcal{L}_{hdir} + \omega_{hpos}\mathcal{L}_{hpos} \\
& + \omega_{len}\mathcal{L}_{len} + \omega_{tang}\mathcal{L}_{tang} + \omega_{cur}\mathcal{L}_{cur}.
\end{aligned}
$$

By utilizing momentum information across the temporal axis, we can provide a better initialization of $\mathbf{S}_n^{t+1}$ given its trajectory and intialize $\mathbf{S}_n^{t+1}$ as

$$
\mathbf{S}_n^{t+1} = 3\mathbf{S}_n^t - 3\mathbf{S}_n^{t-1} + \mathbf{S}_n^{t-2}.
$$

4.2.2 HVH

Background. Similar to MVP, we define volumetric primitives $\mathcal{V}_n = \{\mathbf{t}_n, \mathbf{R}_n, \mathbf{s}_n, \mathbf{V}_n\}$ to model a volume of local 3D space each, where $\mathbf{R}_n \in SO(3), \mathbf{t}_n \in \mathbb{R}^3$ describes the volume-to-world transformation, $\mathbf{s}_n \in \mathbb{R}^3$ are the per-axis scale factors and $\mathbf{V}_n = [\mathbf{V}_c, \mathbf{V}_\alpha] \in \mathbb{R}^{4\times M\times M\times M}$ is a volumetric grid that stores three channel color and opacity information. The volumes are placed on a UV-map that are unwrapped from a head tracked mesh and are regressed from a 2D CNN. Using an optimized BVH implementation, we can efficiently determine how the rays intersect each volume and find hit boxes. For each ray $\mathbf{r}_p(t) = \mathbf{o}_p + t\mathbf{d}_p$, we denote (t_{min}, t_{max}) as the start and end point for ray integration. Then, the differentiable aggregation of those volumetric primitives is defined as:

$$\mathcal{I}_p = \int_{t_{min}}^{t_{max}} \mathbf{V}_c(\mathbf{r}_p(t)) \frac{dT(t)}{dt} dt,$$
$$T(t) = min(\int_{t_{min}}^{t} \mathbf{V}_\alpha(\mathbf{r}_p(t)) dt, 1).$$

We composite the rendered image as $\tilde{\mathcal{I}}_p = \mathcal{I}_p + (1 - \mathcal{A}_p) I_{p,bg}$ where $\mathcal{A}_p = T(t_{max})$ and $I_{p,bg}$ is the background image.

Encoder. The encoder uses the driving signal of a specific point in time and outputs a global latent code $\boldsymbol{z} \in \mathbb{R}^{256}$. We use the tracked guide hairs $\{\mathbf{S}_n^t\}$ and tracked head mesh vertices $\{\boldsymbol{v}_m^t\}$ to define the driving signal. Symmetrically, we learn another decoder in parallel with the encoder in an auto-encoding way that regresses the tracked guide hairs $\{\mathbf{S}_n^t\}$ and head mesh vertices $\{\boldsymbol{v}_m^t\}$ from the global latent code $\boldsymbol{z}$. The architecture of the encoder is an MLP that regresses the parameter of a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}), \boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathcal{R}^{256}$. We use the reparameterization trick from [34] to sample $\boldsymbol{z}$ from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ in a differentiable way.

Hair Volume Decoder. Besides the volumes that are attached to the tracked mesh $\{\boldsymbol{v}_m^t\}$, we define additional hair volume $\mathcal{V}_n^{hair}$ that are associated with guide hair nodes $\mathbf{S}_n^t$. The position $\mathbf{t}_n = \hat{\mathbf{t}}_n + \delta_{\mathbf{t}_n}$, orientation $\mathbf{R}_n = \delta_{\mathbf{R}_n} \cdot \hat{\mathbf{R}}_n$ and scale $\mathbf{s}_n = \hat{\mathbf{s}}_n + \delta_{\mathbf{s}_n}$ of each hair volume are determined by the base hair transformation $(\hat{\mathbf{t}}_n, \hat{\mathbf{R}}_n, \hat{\mathbf{s}}_n)$ and regressed hair relative transformation $(\delta_{\mathbf{t}_n}, \delta_{\mathbf{R}_n}, \delta_{\mathbf{s}_n})$. The base translation $\hat{\mathbf{t}}_n$ of each hair node is directly its position $\mathbf{S}_n^t$. The base rotation $\hat{\mathbf{R}}_n$ is derived from the hair tangential direction and the hair-head relative position. We denote τ_n as the hair tangential direction at position $\mathbf{S}_n^t$ and ν_n' as the direction pointing to the tracked head center starting from S_n^t. Then, the base rotation is $\hat{\mathbf{R}}_n = [\tau_n^T; \rho_n^T; \nu_n^T]$, where $\rho_n = \tau_n \times \nu_n', \nu_n = \rho_n \times \tau_n$.

The geometry of hair can not be simply described by a surface. Therefore, we design a 2D CNN that convolves along the hair growing direction and the rough hair spatial position separately. Specifically, in the each layer of the 2D CNN, we seperate a $k \times k$ filter into two $k \times 1$ and $1 \times k$ filters and apply convolution along two orthogonal directions respectively, similar to [113]. To learn a more consistent hair shape and appearance model, we optimize per-strand hair features $\{f_n^t\}$ that are shared across all time frames besides the temporally varying global latent code $\boldsymbol{z}$. For each node $\mathbf{S_n^t}$ on a hair strand $\mathbf{S}^t$, we assign an unique feature vector f_n^t. The shared per-strand hair features and the temporal varying latent code $\boldsymbol{z}$ are fused to serve as the input to the hair volume decoder, which is shown in Fig. 4.2.

Differentiable Volumetric Raymarching of 3D Scene Flow. Learning a volumetric scene representation by multi-view photometric information is sufficient for high fidelity rendering and novel view synthesis. However, it is challenging for the

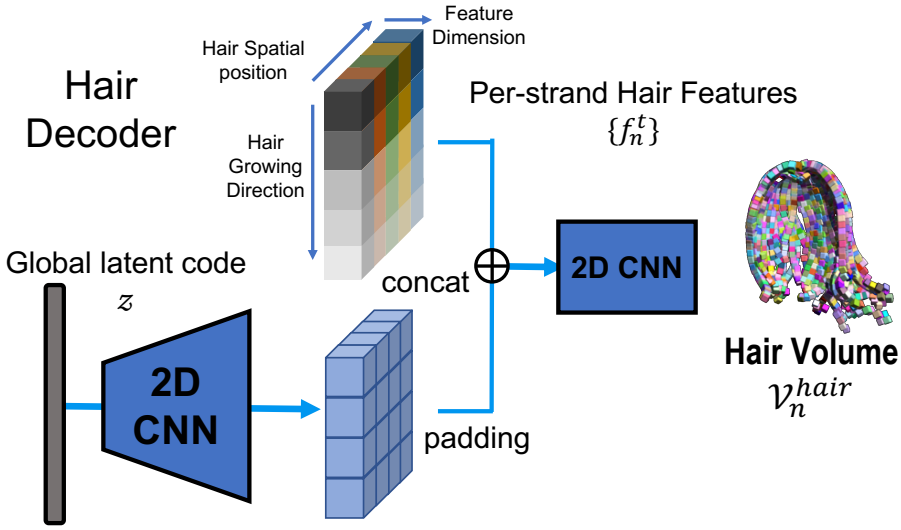

Figure 4.2: **Architecture of the hair decoder.** The hair decoder takes both the global latent code z and the per-strand hair features $\{f_n^t\}$ as inputs. z is first deconvolved into a 2D feature tensor. It is then padded and concatenated with $\{f_n^t\}$. In the following operation, the 2D convolution layers are applied along the hair growing direction and the hair spatial position seperately.

model to reason about motion given the limited supervision and the results have poor temporal consistency, especially on unseen sequences. To better enforce temporal consistency, we develop a differentiable volumetric ray marching algorithm of 3D scene flow which enables training via multi-view 2D optical flow.

Given the transformations of each primitive as $(\mathbf{t}_n, \mathbf{R}_n, \mathbf{s}_n)$, we express the coordinate of each node on a volumetric grid at frame u as $\mathbf{V}_{xyz}^u = \mathbf{s}_t \mathbf{R}_t \mathbf{V}_{tpl} + \mathbf{t}_n$, where $\mathbf{V}_{tpl}$ are the coordinates of a 3D mesh grid ranging between $[-1, 1]$. Given that the 3D scene flow from frame u to $u + \delta$ can be expressed by each volumetric primitives as $\{\delta \mathbf{V}_{xyz}^{u,u+\epsilon} = \mathbf{V}_{xyz}^{u+\epsilon} - \mathbf{V}_{xyz}^u\}$ and rendered into 2D flow as:

$$\mathcal{I}_{p,flow}^{u,u+\delta} = \int_{t_{min}}^{t_{max}} (\delta \mathbf{V}_{xyz}^{u,u+\epsilon}(\mathbf{r}_p(t))) \frac{dT(t)}{dt} dt,$$
$$T(t) = min(\int_{t_{min}}^{t_{max}} \mathbf{V}_{\alpha}^u(\mathbf{r}_p(t)) dt, 1).$$

Training Objectives. We train our model in an end-to-end manner with the following loss:

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{pho} + \lambda_{flow}\mathcal{L}_{flow} + \lambda_{geo}\mathcal{L}_{geo} \\ &+ \lambda_{vol}\mathcal{L}_{vol} + \lambda_{cub}\mathcal{L}_{cub} + \lambda_{KL}\mathcal{L}_{KL}.\end{aligned}$$

The first term $\mathcal{L}_{pho}$ is the photometric loss that compares the difference between the rendered image $\tilde{\mathcal{I}}_p$ and ground truth image I_p on all sampled pixels $p \in \mathcal{P}$,

$$\mathcal{L}_{pho} = \sum_{p \in \mathcal{P}} ||I_{p,gt} - \tilde{\mathcal{I}}_p||_2^2.$$

The second term $\mathcal{L}_{flow}$ aims to enforce temporal consistency of volumetric primitives from frame u and its adjacent frame $u + \epsilon$ by minimizing the projected 2D flow and ground truth optical flow $I_{p,flow}^{u,u+\epsilon}$,

$$\mathcal{L}_{flow} = \sum_{p \in \mathcal{P}} \mathcal{A}_p ||I_{p,of}^{u,u+\epsilon} - \mathcal{I}_{p,flow}^{u,u+\epsilon}||_2^2,$$

where $\epsilon \in \{-1, 1\}$. It is important to note that we use $\mathcal{A}_p$ to mask out the background part and we do not back propagate the errors from $\mathcal{L}_{flow}$ to $\mathcal{A}_p$ in order to get rid of the background noise in optical flows. To better enforce hair and head primitives moving with the tracked head mesh and guide hair strands, $\mathcal{L}_{geo}$ is designed to measure the difference between the mesh/strand vertices and their corresponding regressed value.

$$\mathcal{L}_{geo} = \sum_n ||\boldsymbol{S}_n^t - \boldsymbol{S}_{n,gt}^t||_2^2 + \sum_m ||\boldsymbol{v}_m^t - \boldsymbol{v}_{m,gt}^t||_2^2,$$

where $\boldsymbol{S}_n^t$ and $\boldsymbol{v}_m^t$ are the coordinate of the nth node of the tracked guide hair and tracked head mesh at frame t and the X_{gt} denotes the corresponding ground truth value.

We also add several regularization terms to inform the layout of the volumetric primitives:

$$\begin{aligned}\mathcal{L}_{vol} &= \sum_{i=1,\cdots,N_p} \prod_{j \in \{x,y,z\}} s_i^j, \\ \mathcal{L}_{cub} &= \sum_{i=1,\cdots,N_p} ||max(s_i^x, s_i^y, s_i^z) - min(s_i^x, s_i^y, s_i^z)||,\end{aligned}$$

where N_p stands for the total number of volumetric primitives and s_i^x, s_i^y, s_i^z are the three entries of each volumetric primitive's scale $\boldsymbol{s}_j$. The two regularization terms aim to prevent each primitive from growing too big while preserving the aspect ratio so that they remain approximately cubic. The last term is the Kullback-Leibler divergence loss $\mathcal{L}_{KL}$ which makes the learnt distribution of latent code $\boldsymbol{z}$ smooth and enforces similarity with a normal distribution $\mathcal{N}(0, 1)$.

4.3 Experiments

4.3.1 Dataset

For each video recorded with our multi camera system, we split the them by the motions performed (like nodding and shaking of the head) and hold out the last $\frac{1}{4}$ of each motion for testing drivable animation. This results in roughly 300 frames for training sequence and 100 frames for testing sequence. Additionally, on the training sequence, we hold out 7 cameras that are distributed around the rear and side view of the head. The captured images are downsampled to 1024×667 resolution for training and testing. We train our model exclusively on the training portion of each sequence with $m = 93$ training views.

We use a multi-camera system with around 100 synchronized color cameras that produces 2048×1334 resolution images at 30 Hz. The cameras are focused at the center of the capture system and distributed spherically at a distance of one meter to provide as many viewpoints as possible. Camera intrinsics and extrinsics are calibrated in an offline process. We captured three sequences of different hair styles and hair motions. In the first sequence, we have one actor with a short high pony tail performing nodding and rotating. In the second sequence, we have one actor with a curly long releasing style hair and leaning her head towards four directions(left, right, up and down) and rotating. In the third sequence, we have one actor with a long high pony tail performing nodding and rotating.

Diversity in hairstyles: Given that the main focus of this work is dynamic hair capture and tracking, we selected several hairstyles with a certain level of diversity, like long curly open hair, mid-length fluffy straight pony tail, and long curly pony tail, that exhibit complex dynamic behavior where hair does not move rigidly with the head—hence are particularly well-suited for analyzing the performance of the proposed approach. Regarding generalization, the 3D scene flow formulation and the hair decoder are agnostic to specific hair structure and color; the hair tracking algorithm depends on artist prepared guide strands and, together with the optical flow, requires sufficient contrast for hair strands and background. Given its strand-based nature, our method might not be suitable for specific hairstyles like buzz

cut or afro-textured hair, where it is challenging to create the initialization of the strands. However, we want to point out that our 3D scene flow formulation, which is agnostic to hair style, alone already improves MVP (as shown in the experiments).

4.3.2 Baselines

We compare against several volume-based or implicit function based baseline methods [40, 47, 95] for spatio-temporal modeling.

MVP [47] presents an efficient 4D representation for dynamic scenes with humans which is capable of doing animation and novel view synthesis. It combines explicitly tracked head mesh with volumetric primitives to model the human appearance and geometry with better completeness. The volumetric primitives can be aligned onto an unwrapped 2D UV-map from a tracked head mesh and can be regressed from a 2D convolutional neural network that leverages shared spatially computation. Similar to Neural Volumes [46], a differentiable volumetric ray marching algorithm is designed to render 2D rgb images on MVP in real time. We use $N_p = 4096$ volumetric primitives with a voxel resolution $8 \times 8 \times 8$ on each sequence with a ray marching step size around $dt = 1mm$. We use a global latent size of 256.

Non-rigid NeRF [95] presents an implicit function based representation for dynamic scene reconstruction and novel view synthesis based on NeRF [61]. It utilizes a hierarchical model by disentangling a dynamic scene into a canonical frame NeRF and its corresponding deformation field which is parameterized by another MLP. In our experiments, we use 128 sampling points for both coarse and fine level sampling. We use the original implementation from the authors here. We train different models for each sequences and each model is trained for at least 300k iterations until convergence.

NSFF [40] is another implicit function based representation for dynamic scenes that is also based on NeRF [61]. It learns a per-frame NeRF that is additionally conditioned on the time index. It brings optical flow as additional supervision and learns a 3D scene flow in parallel with the per-frame NeRF for enforcing temporal consistency. NSFF is able to perform both spatial and temporal interpolation on a given video sequence. We use a setting of 256 sampling points in our experiments, using [36] as a substitute for generating optical flow. We use the original implementation from the authors here. We train different models for each sequences and each model is trained for at least 300k iterations until convergence.

4.3.3 Training Details

For both tracking optimization and HVH training, We deploy Adam [33] for optimization. For hair tracking, we use a learning rate of 1. We set the weighting

coefficients of each losses as $\omega_{hdir} = 3$, $\omega_{hpos} = 1$, $\omega_{len} = 3$, $\omega_{tang} = 3$ and $\omega_{cur} = 1e4$. For each time step, 100 iterations are taken for optimization to solve the possible hair strands at next frame out. For HVH, we set weighting parameters for each objective as $\lambda_{flow} = 1$. $\lambda_{geo} = 0.1$, $\lambda_{vol} = 0.01$, $\lambda_{cub} = 0.01$ and $\lambda_{KL} = 0.001$. All models are trained with approximately 100-150k iterations. We use a latent code size of 256 and per-strand hair code size of 256, raymarching step size around $dt = 1mm$ and around $N_p = 5500$ volumetric primitives with a voxel resolution $8 \times 8 \times 8$ for each sequence depending on the number of guide hairs. For each sequence, we have roughly 30 strands for guide hair and we sample 50 points on each strands.

4.3.4 Novel View Synthesis

	Seq01			Seq02			Seq03		
	MSE	SSIM	PSNR	MSE	SSIM	PSNR	MSE	SSIM	PSNR
PFNeRF	51.25	0.9269	31.16	103.41	0.8659	28.15	76.59	0.9000	29.50
NSFF	50.13	0.9346	31.21	90.06	0.8885	28.75	83.18	0.8936	29.1
NRNeRF	56.78	0.9231	30.78	132.16	0.8549	27.13	79.83	0.8987	29.33
MVP	47.54	0.9476	31.6	77.23	0.9088	29.62	73.78	0.9224	29.66
Ours	**41.89**	**0.9543**	**32.17**	**59.84**	**0.9275**	**30.69**	**71.58**	**0.9314**	**29.81**

Table 4.1: **Novel view synthesis**. We compare our method with both NeRF stemmed methods like NSFF [40], NRNeRF [95] and a per-frame NeRF (PFNeRF) baseline, and a volumetric method like MVP [47]. As we can see, our methods achieves the best performance on image reconstruction metrics.

We show both qualitative and quantitative comparisons with other methods [40, 47, 95] on the novel view synthesis task. In Tab. 4.1, we show the mean squared error (MSE), SSIM and PSNR between predicted images and ground truth images from the novel views of the training sequences. Qualitative results are shown in Fig. 4.3. Our method has smaller image prediction errors and is able to generate sharper results, especially on the hair regions.

We show a larger version of comparison figure between different methods in Figure 4.4. For completeness, we also include visualizations from a perframe NeRF model which takes a perframe temporal code as input liker non-rigid NeRF [95].

4.3.5 Ablation Studies

Temporal consistency. To test the effects of the temporal consistency and the tracked guide hair, we also conduct a novel view synthesis task on the test portion

	Seq01			Seq02			Seq03		
	MSE	SSIM	PSNR	MSE	SSIM	PSNR	MSE	SSIM	PSNR
MVP	47.54	0.9476	31.6	77.23	0.9088	29.62	73.78	0.9224	29.66
MVP w/ $\mathcal{L}_{flow}$	46.49	0.9473	31.69	71.07	0.9107	29.93	75.13	0.9240	29.58
Ours w/o $\mathcal{L}_{flow}$	43.82	0.9508	31.99	65.98	0.9186	30.27	69.97	0.9359	29.93
Ours	**41.89**	**0.9543**	**32.17**	**59.84**	**0.9275**	**30.69**	**71.58**	**0.9314**	**29.81**
MVP	75.68	0.9200	29.49	85.10	0.9039	29.62	83.76	0.9086	29.16
MVP w/ $\mathcal{L}_{flow}$	67.86	0.9276	30.00	83.11	0.9037	29.93	80.96	0.9086	29.16
Ours w/o $\mathcal{L}_{flow}$	71.90	0.9223	29.74	72.74	0.9137	30.27	78.34	0.9198	29.44
Ours	**65.96**	**0.9280**	**30.09**	**67.75**	**0.9208**	**30.69**	**75.66**	**0.9222**	**29.57**

Table 4.2: **Novel view synthesis**. We further compare our method and different variants of our methods with MVP [47] on novel views of both seen (top) and unseen (bottom) sequences. We find that using the optical flow to enforce the temporal consistency leads to improvement on both MVP [47] and our method, while the best results are achieve when coarse level guide hair tracking is combined with fine level flow optimization.

of our captured sequence. Note that our model is not trained using any part of the test sequence data. In Tab. 4.2, we report MSE, SSIM, PSNR from different variants of our method with MVP on novel views of both seen and unseen sequences. As we can see, having the coarse level guide hair strands tracked and without flow supervision gives us better rendering quality. With flow supervision, the results are improved further. This improvement is because the tracking information helps the volumetric primitives to better localize the hair region with higher consistency. While the improvement for seen motions is relatively small, both our model and MVP are notably improved for unseen sequences with novel hair motion when flow supervision is added. Rendering results on unseen sequences are shown in Fig. 4.8. In Fig. 4.9, we visualize the volumetric primitives of the hairs of our model with and without flow supervision. Including flow supervision produces notably better disentanglement between the hair and shoulder.

We show a bigger version of rendering results on unseen sequence in Figure 4.10.

Hair Decoder structure. As part of the hair decoder ablation, we compare our method with a naive decoder that uses the same volume decoder as MVP [47] for hair volumes. There are two major differences: 1) the naive decoder does not take the per-strand hair feature as input; 2) The design of the naive decoder does not take into account the hair specific structure where it regresses the same slab as for head tracked mesh and we take the first N_{hair} volumes as the output. In this way, the naive decoder discards all intrinsic geometric structural information while doing convolutions in each layers. We show the hair volumes layout in Figure 4.11. In the

Figure 4.3: **Comparison on novel view synthesis between different methods.** We compare our method on novel view synthesis with different volumetric methods like a perframe time conditioned NeRF model, NSFF [40], NRNeRF [95] and MVP [47]. Rendering results on three different subjects with different hairstyles are shown. Interesting parts of hair with details are highlighted using a red bounding box. As we can see, our method is capable of generating a consistent global shape while also capturing enough details.

naive design, the hair strands are randomly squeezed into a square UV-map which could break the inner connections of each hair. In our design, we groom the hair strands into the their directions which could preserve the hair specific geometric structure. We compare different designs of decoder on Seq01. As in Table 4.3, our hair structure aware decoder produces a smaller image reconstruction error and better SSIM, a result of inductive bias of the designed hair decoder.

We additionally compare two different designs of the hair decoder where we do late and early fusion of the per-strand hair feature and the global latent feature.

Figure 4.4: **Comparison on novel view synthesis between different methods.** We compare our method with different volumetric methods including a perframe time conditioned NeRF model, NSFF [40], NRNeRF [95] and MVP [47].

decoder	MSE	SSIM	PSNR
naive	45.68/75.15	0.9549/0.9220	31.83/29.54
early fus.	43.75/71.08	0.9533/0.9259	31.97/29.82
late fus.	41.89/65.96	0.9543/0.9280	32.17/30.09

Table 4.3: **Decoder structure.** We compare different designs of the hair decoder. We report all metrics on both training and testing and we use a to separate them where on the left are the results of novel synthesis on training sequence.

We show two different designs in Figure 4.12. Table 4.3 shows that the late fusion model performs better than early fusion model. This could be because the late fusion model transfers the 1d global latent code into a spatially varying feature tensor which is a more expressive form of feature representation.

Hair tracking analysis. We first study the impact of different objectives $\mathcal{L}_{len} + \mathcal{L}_{tang}$ and $\mathcal{L}_{cur}$ in hair tracking. As in Fig. 4.13, when both $\mathcal{L}_{cur}$ and $\mathcal{L}_{len} + \mathcal{L}_{tang}$

are applied, the tracking results are more smooth and without kinks. We observe that, when using the loss $\mathcal{L}_{len} + \mathcal{L}_{tang}$ as the only regularization term, the length of each hair strand segments are already preserved but could cause some kinks without awareness of the correct hair strand curvatures. $\mathcal{L}_{cur}$ itself does not help and exaggerates the error when the hair strand length is not correct, but yields smooth results when combined with $\mathcal{L}_{len} + \mathcal{L}_{tang}$. This is because curvature computation is agnostic to absolute length of the hair and only controls the relative length ratio.

We show the impact of different initialization for hair tracking in Fig. 4.14. When no momentum information from previous frames is used, there is more obvious drifting on some of the strands happening, while the drifting is less severe when we take advantage of the motion information from previous frames.

In Figure 4.15, we plot different hair properties over time. We report four different metrics describing how well the tracked hairs fit the per-frame reconstruction and how well it preserves its length and curvature. In the first two rows, we report the MSE between the tracked hair and the tracked hair at first frame in terms of curvature and length. In the last two rows, we report the cosine distance between the direction of each nodes on the tracked guide hair and the direction of its neighbor from the reconstruction and the Chamfer distance between the tracked guide hair nodes and the reconstruction. As we can see the length and curvature are relatively preserved across frames and the affinity between the per-frame reconstruction and the tracked guide hair is relatively high.

Visualization of Flow: Please see Figure 4.16 for a visualization of the rendered flow from our representation. Compared to the optical flow from [36], our rendered 2D flow has less noise on the background. This is because that we only define our 3D scene flow on the volumetric primitives instead of the whole space. With the help of the coarse level geometry like the hair strands and head tracked mesh, the scene flow of most part of the empty space will naturally be zero. This could help us eliminate the noise from the background optical flow to certain degree.

Run Time Analysis. We report the rendering time of one iamge at resolution 1024×667 for each methods here. MVP [47] takes 0.223s. Ours takes 0.254s. NSFF takes 28.68s. NRNeRF [95] takes 41.29s. All tests are conducted under a single Nvidia Tesla V100 GPU.

4.4 Video Results

Please see all the video results on this page[1].

[1] `https://ziyanw1.github.io/hvh/`

4.5 Applications and Limitations

One major application that is enabled by our neural volumetric scene representation is novel view synthesis as we have shown in Sec. 4.3.4. Our neural volumetric representation is also animatable with a sparse driving signals like guide hair strands. Given that we have explicitly modeled hair in the form of guide strands, our method allows modifying the guide hairs directly. In Fig. 4.17, we show four snapshots of different configurations of hair positions.

There are several limitations of our work which we plan to address in the future: 1) Our method requires the help from artist to prepare guide hair at the first frame and some flyaway hair might be excluded. 2) We currently do not consider physics based interactions between hair and other objects like the shoulder or the chair. 3) Although we achieved certain level of disentanglement between hair and other objects without any human labeling, it is still not perfect. We only showed results on blonde hair which could be better distinguished from a dark background. Our method might be limited by other hairstyles like buzz cut or afro-textured that are hard for artist to prepare guide hair. Future directions like incorporating a physics aware module or leveraging additional supervision from semantic information for disentanglement could be interesting.

4.6 Discussion

In this chapter, we present a hybrid neural volumetric representation for hair dynamic performance capture. Our representation leverages the efficiency of guide hair representation in hair simulation by attaching volumetric primitives to them as well as the high DoF of volumetric representation. With both hair tracking and 3D scene flow refinement, our model enjoys better temporal consistency. We empirically show that our method generates sharper and higher quality results on hair and our method achieves better generalization. Our model also supports multiple applications like drivable animation and hair editing.

Figure 4.5: **Rendering results on subject 3.** We show more rendering results under two novel views on subject 3. From left to right, we show results of a perframe NeRF, NRNeRF [95], NSFF [40], MVP [47], ours and ground truth.

Figure 4.6: **Rendering results on subject 2.** We show more rendering results under two novel views on subject 3. From left to right, we show results of a perframe NeRF, NRNeRF [95], NSFF [40], MVP [47], ours and ground truth.

Figure 4.7: **Rendering results on subject 1.** We show more rendering results under two novel views on subject 3. From left to right, we show results of a perframe NeRF, NRNeRF [95], NSFF [40], MVP [47], ours and ground truth.

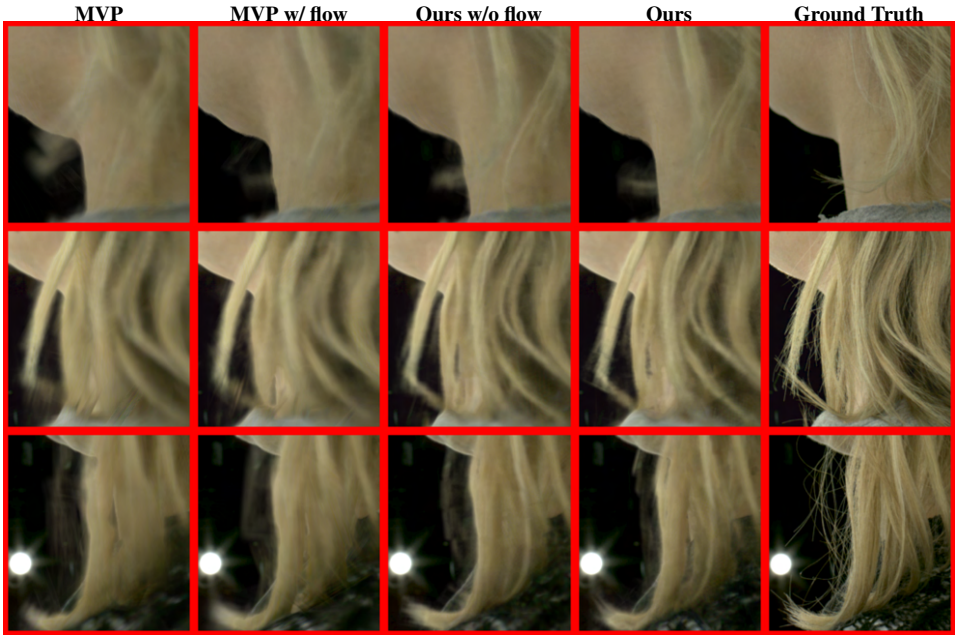

Figure 4.8: **Ablation of temporal consistency.** We compare our method and MVP w/ and w/o flow supervision. With flow supervision, better temporal consistency and generalization for unseen sequence can be observed.

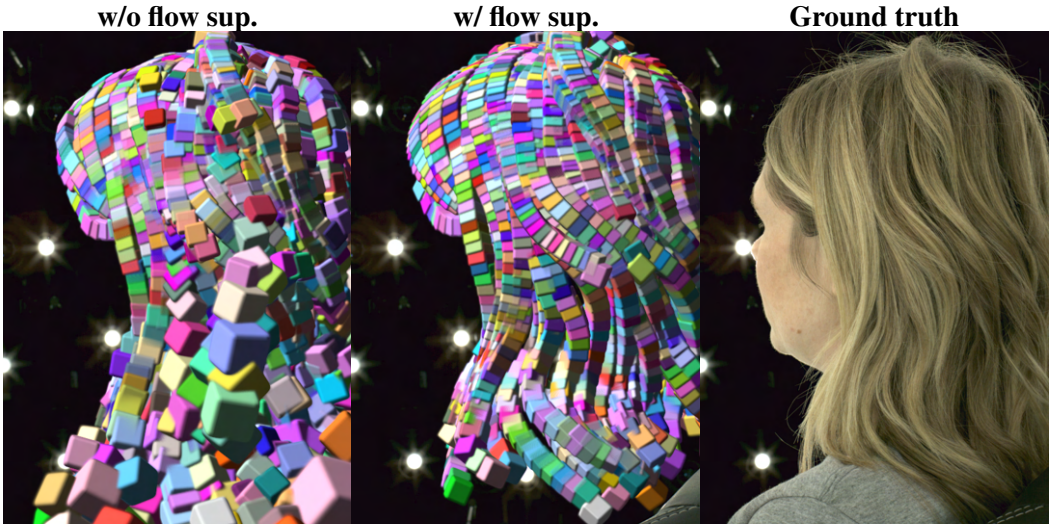

Figure 4.9: **Ablation on flow supervision.** We further compare the volumetric primitives of the models w/ and w/o flow supervision. We see that the model with additional flow supervision yields a consistent and reasonable shape for hair and yields better hair shoulder disentanglement.

Figure 4.10: **Ablation of temporal consistency.** We compare MVP [47] and ours with different variations.

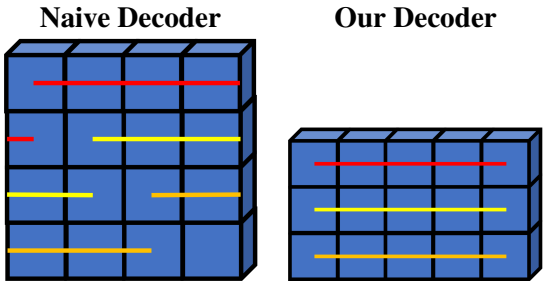

Figure 4.11: **Hair volumes layout.** We show the hair volume layout of both naive decoder and ours.

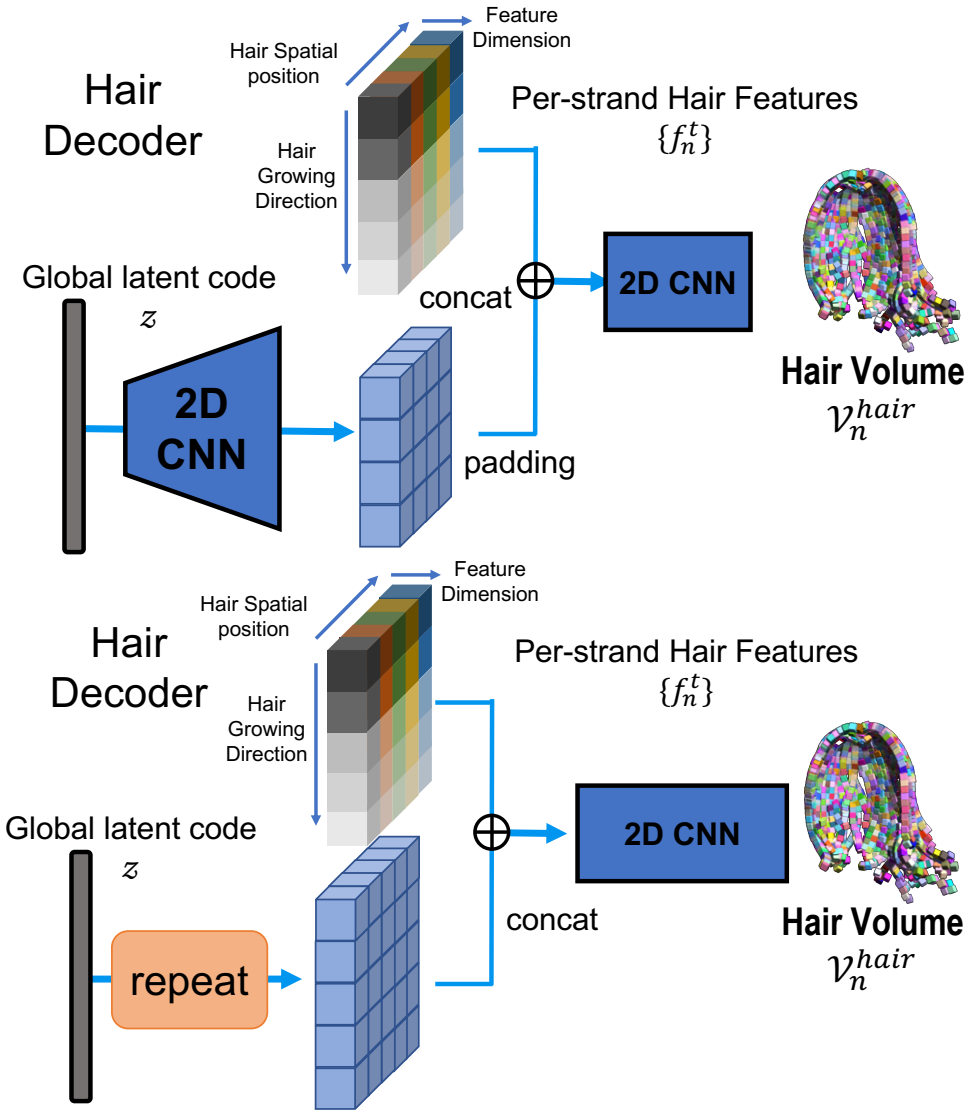

Figure 4.12: **Architecture of the hair decoder.** We show late fusion on the top and early fusion on the bottom. The late fusion model first deconvolves the 1D global latent code into a 2D feature map and then concatenate it with the per-strand hair features. A 2D CNN is used afterwards to generate the hair volumes. The early fusion model first repeat the 1D global latent vector spatially and then concatenate the repeated feature map with per-strand hair features. The concatenated features are than fed into a deeper 2D CNN to generate the hair volumes.

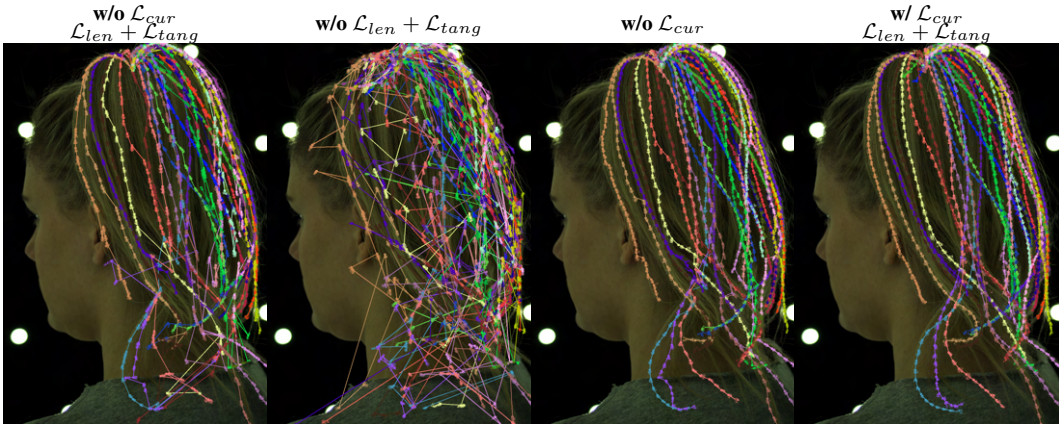

Figure 4.13: **Effects of $\mathcal{L}_{len} + \mathcal{L}_{tang}$ and $\mathcal{L}_{cur}$.** We show how the shape and curvature of tracked hair strands are preserved with both $\mathcal{L}_{len} + \mathcal{L}_{tang}$ and $\mathcal{L}_{cur}$. Point on the same strand are visualized in the same color and adjacent points are connected with line in the same color. When no regularization on hair strand geometry is applied, some part of the hair strand get stretched or become zigzag. When only the second order regularization $\mathcal{L}_{cur}$ is applied, we find the results become more unstable. When first order regularization $\mathcal{L}_{len} + \mathcal{L}_{tang}$ is applied, the tracked hair strand become more stable but zigzags still persist. When all terms are applied, we get the most smooth result. This suggest that all regularization terms are supposed to be applied together.

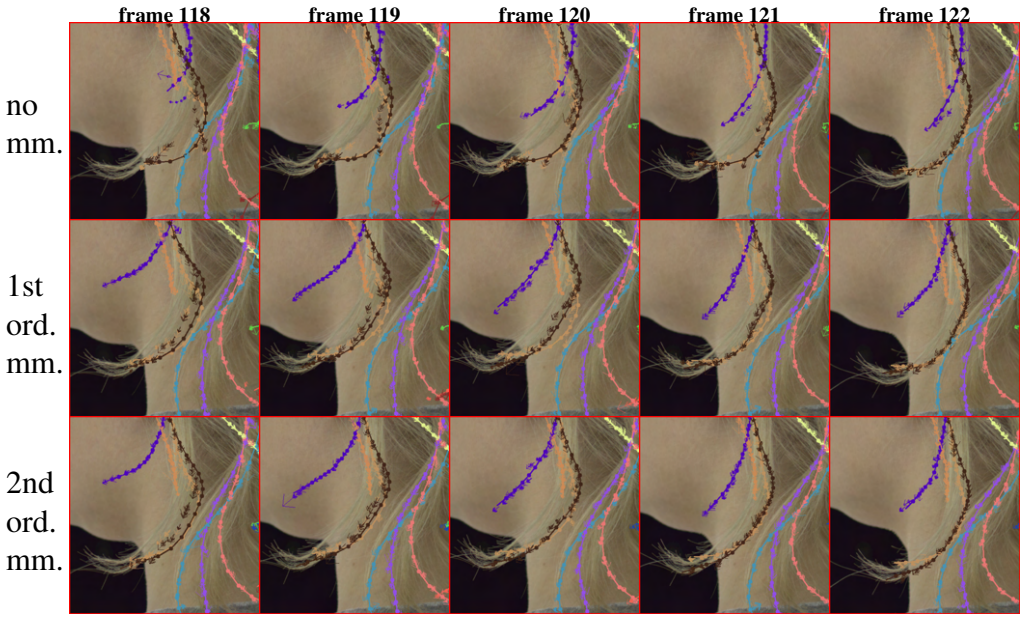

Figure 4.14: **Ablation of different initialization in hair tracking.** We show tracking results of our methods with different initializations. From top to bottom, we use no momentum information, first and second order momentum information for tracking initialization. Please note the brown and orange strands. As we can see, the hairs are better tracked when we utilize the dynamic information from previous frames. Better view in color version.

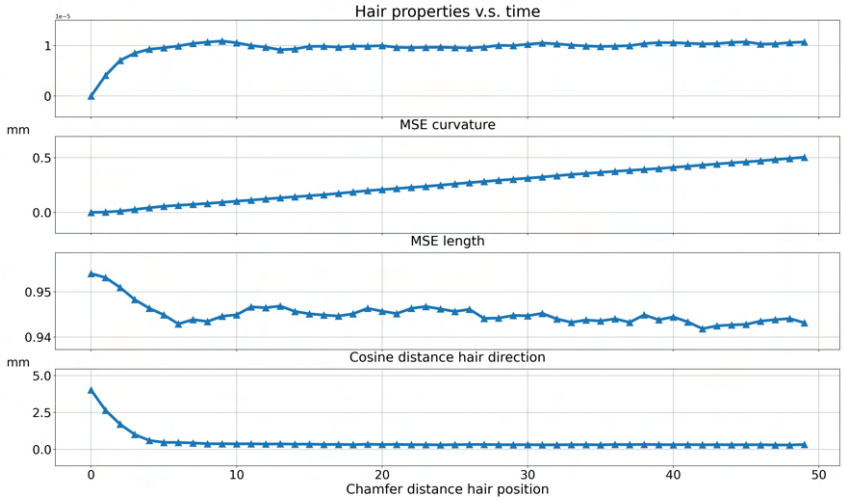

Figure 4.15: **Plot of tracked hair properties v.s. time.** As we can see, the hair properties like length and curvature are not changing too much across time and hair Chamfer distance are relatively small.

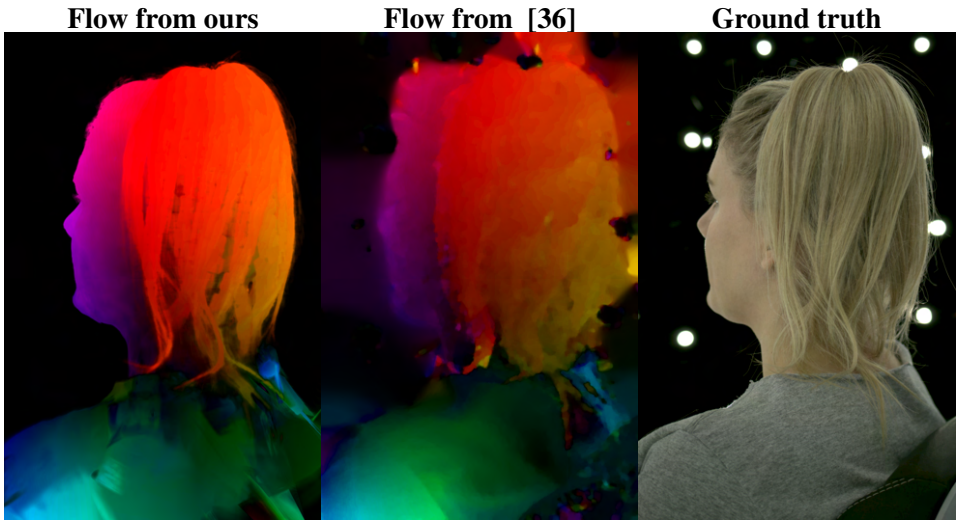

Figure 4.16: **Visualization of flow.** We show the rendered 3D scene flow into 2D flow in the first column and the openCV optical flow [36] in the second column. The last column shows the ground truth image as reference.

Figure 4.17: **Hair position editing.** We create a new animation by direct editing on the guide hair strands. As the guide hair provide a tangible interface to control the hair part, we can directly drive the volumes of hair by adding motion to the guide hair like lifting it up to create new animation.

Chapter 5

NeuWigs: A Neural Dynamic Model for Volumetric Hair Capture and Animation

5.1 Introduction

The ability to model the details of human hair with high fidelity is key to achieving realism in human avatar creation because hair establishes part of our personal identity.One big challenge towards that goal is how to capture the hair dynamics. While modern capture systems reconstruct the hair geometry and appearance from a sparse and discrete set of real world observations with high fidelity, the problem of tracking is not trivially solved by that as not temporal alignment between those reconstructions is established.

In the previous chapter 4, we designed a forward tracking algorithm based on data-driven prior of flow. Although many hand-crafted regularizers on hair shape and curvatures are applied, the tracking algorithm is not robust to long sequence as errors accumulate along the forward pass. Furthermore, it relies on artist input to prepare the guide hair strand as an initialization. This design becomes problematic when it is hard for the artist to prepare guide hair for certain hairstyle or register different states of the same hairstyle.

Another challenge is how to generate realistic hair dynamics. Generating realistic dynamics given only the head motion is similarly hard because the motion of hair is not solely controlled by the head position but also influenced by gravity and inertial forces. From a control perspective, hair does not respond linearly to the head position and a zero-order system is inadequate for modeling hair dynamics. To achieve that, we need to go beyond reconstructing and tracking to create a *controllable* dynamic hair model using captured data.

In conventional animation techniques, hair geometry is created by an artist manually preparing 3D hair grooms. Motion of the 3D hair groom is created by a physics simulator where an artist selects the parameters for the simulation. This process requires expert knowledge. In contrast, data-driven methods aim to achieve hair capture and animation in an automatic way while preserving metric photorealism. Most of the current data-driven hair capture and animation approaches learn to regress a dense 3D hair representation that is renderable directly from per-frame driving signals, without modeling dynamics.

However, there are several factors that limit the practical use of these data-driven methods for hair animation. First of all, these methods mostly rely on sophisticated driving signals, like multi-view images [46, 69], a tracked mesh of the hair [47], or tracked guide hair strands [102], which are hard to acquire. Furthermore, from an animation perspective, these models are limited to rendering hair based on observations and cannot be used to generate novel motion of hair. Sometimes it is not possible to record the hair driving signals at all. We might want to animate hair for a person wearing accessories or equipment that (partially) obstructs the view of their hair, for example VR glasses; or animate a novel hair style for a subject.

Figure 5.1: **Animation from Single View Captures.** Our model can generate realistic hair animation from single view video based on head motion and gravity direction. Original captures of subjects wearing a wig cap are shown in red boxes.

To address these limitations of existing data-driven hair capture and animation approaches, we present a neural dynamic model that is able to animate hair with high fidelity conditioned on head motion and relative gravity direction. By building such a dynamic model, we are able to generate hair motions by evolving an initial hair state into a future one, without relying on per-frame hair observation as a driving signal. We utilize a two-stage approach for creating this dynamic model: in the first stage, we perform state compression by learning a hair autoencoder from multi-view video captures with an evolving tracking algorithm. As adjacent hair states are temporally continuous, we believe that such property can be preserved when they are encoded into an embedding space and tracking between them are automatically discovered through the encoding learning. Our method is capable of capturing a temporally consistent, fully renderable volumetric representation of hair from videos with both head and hair. Hair states with different time-stamps are parameterized into a semantic embedding space via the autoencoder. In the second

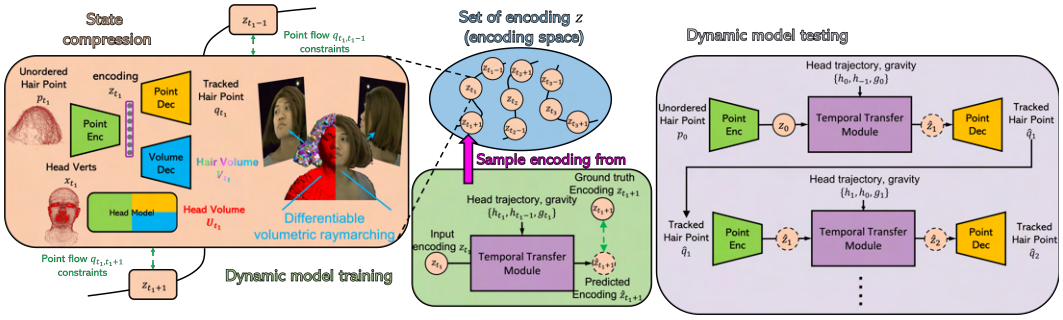

Figure 5.2: **Method Overview.** Our method is comprised of two stages: state compression and dynamic modeling. In the first stage, we train an autoencoder for hair and head appearance from multiview RGB images using differentiable volumetric raymarching; at the same time we create an encoding space of hair states. In the dynamic modeling stage, we sample temporally adjacent hair encodings to train a temporal transfer module ($\mathtt{T^2M}$) that performs the transfer between the two, based on head motion and head-relative gravity direction.

stage, we sample temporally adjacent pairs from the semantic embedding space and learn a dynamic model that can perform the hair state transition between each state in the embedding space given the previous head motion and gravity direction. With such a dynamic model, we can perform hair state evolution and hair animation in a recurrent manner which is not driven by existing hair observations. As shown in Fig 5.1, our method is capable of generating realistic hair animation with different hair styles on single view captures of a moving head with a bald cap. In summary, the contributions of this work are

- We present NeuWigs, a novel end-to-end data-driven pipeline with a volumetric autoencoder as the backbone for real human hair capture and animation, learnt from multi-view RGB images.
- We learn the hair geometry, tracking and appearance end-to-end with a novel autoencoder-as-a-tracker strategy for hair state compression, where the hair is modeled separately from the head using multi-view hair segmentation.
- We train an animatable hair dynamic model that is robust to drift using a hair state denoiser realized by the 3D autoencoder from the compression stage.

5.2 Method

Our method for hair performance capture and animation consists of two stages: state compression and dynamic modeling (see also Fig. 5.2). The goal of the first

stage is to perform dynamic hair capture from multi-view video of a head in motion. To be more specific, in this stage, we aim to distill a 3D renderable representation of hair from multi-view images at each frame into an embedding space. To achieve that, we train a volumetric autoencoder in a self-supervised manner to model the hair geometry, tracking and appearance. The output of this model is a set of tracked hair point clouds p_t, their corresponding local radiance fields in the form of volumetric primitives $\boldsymbol{V}_t$ and a compact embedding space that is spanned by the 1D hair state encoding $\boldsymbol{z}_t$. In the second stage, we perform modeling of hair dynamics based on the hair capture from the first stage. The goal of this stage is to create a controllable, self-evolving representation of hair without relying on online observations of hair. We achieve this by learning a neural network to regress the next possible hair state, which is conditioned on the previous hair state as well as the previous head motion and head-relative gravity direction. Equipped with the hair encoding space acquired from the first stage, we can train that model in a supervised manner by simply sampling data pairs of temporally adjacent hair states from the encoding space. Using both stages, we can perform dynamic hair animation at test time given an initialization using a recurrent strategy, without relying on direct observations of hair as a per-frame driving signal.

5.2.1 State Compression

We assume that multi-view image captures I_{cam_i} with their corresponding calibrated cameras are given. We denote the extrinsics of each camera i as $\mathbf{R}_i$ and $\mathbf{t}_i$. We then run l-MVS [63] and non-rigid tracking [106] to obtain the per-frame hair reconstructions $p_t \in \mathbb{R}^{N_{p_t} \times 3}$ and head tracked vertices $x_t \in \mathbb{R}^{N_{x_t} \times 3}$ at time frame t, where N_{p_t} and N_{x_t} denote the size of each. The p_t and x_t together serves as a coarse representation for the hair and head. Different from the head vertices, here the p_t represent an unordered set of hair point clouds. Due to the difference between hair and head dynamic patterns, we model them separately by training two different volumetric autoencoders. For the head model, we use an autoencoder to regress the volumetric texture in an unwrapped UV layout conditioned on the tracked head mesh, similar to [47, 102]. For the hair model, we optimize the hair volumetric texture and its tracking simultaneously. To better enforce the disentanglement between hair and head, we attach head volumes only to the head mesh and hair volumes only to the hair point clouds. Moreover, we use a segmentation loss to constrain each of them to only model the texture of their assigned category of hair or head.

Autoencoder as a Tracker. Learning to track hair in a supervised manner with manual annotation is infeasible. We automatically discover the hair keypoints as well as their tracking information by optimizing a variational autoencoder (VAE) [34] in a semi-supervised manner. By doing autoencoding on hair point clouds, we find

that the VAE representing the hair point cloud can automatically align hair shapes along the temporal axis and is capable of tracking through both long and discontinuous hair video segments such as captures of different hair motions.

The input to the point encoder $\mathcal{E}$ is the point coordinates of the unordered hair point cloud p_t. Given its innate randomness in terms of point coverage and order, we use PointNet [74] to extract the corresponding encoding $\boldsymbol{z_t} \in \mathbb{R}^{256}$. Besides being agnostic to the order of p_t, it also can process varying numbers of points and aggregate global information from the input point cloud. The point decoder $\mathcal{D}$ is a simple MLP that regresses the coordinate and point tangential direction of the tracked point cloud $q_t \in \mathbb{R}^{N_{prim} \times 3}$, $\boldsymbol{dir}(q_t) \in \mathbb{R}^{N_{prim} \times 3}$ and $s_t \in \mathbb{R}^{N_{prim}}$ from $\boldsymbol{z_t}$, where N_{prim} is the number of tracked hair points. We denote $\boldsymbol{dir}(x)$ as the tangential direction of x. s_t is a per-point scale factor which will be used later. We optimize the following loss to train the point autoencoder:

$$\mathcal{L}_{geo} = \mathcal{L}_{cham} + \omega_{temp}\mathcal{L}_{temp} + \omega_{KL}\mathcal{L}_{KL}.$$

The first term is the Chamfer distance loss which aims to align the shape of tracked point cloud q_t to p_t:

$$\begin{aligned}\mathcal{L}_{cham} = ||q_t - \mathtt{N}_{q_t,p_t}||_2 - \boldsymbol{cos}(\boldsymbol{dir}(q_t), \boldsymbol{dir}(\mathtt{N}_{q_t,p_t})) \\ +||p_t - \mathtt{N}_{q_t,p_t}||_2 - \boldsymbol{cos}(\boldsymbol{dir}(p_t), \boldsymbol{dir}(\mathtt{N}_{q_t,p_t})),\end{aligned}$$

where $\boldsymbol{cos}(\cdot,\cdot)$ is the cosine similarity and $\mathtt{N}_{x,y} \in \mathbb{R}^{N_x \times 3}$ are the coordinates of the nearest neighbor of each point of x in y. To further enforce temporal smoothness, we use point flow $\overrightarrow{fl}(p_t)$ and $\overleftarrow{fl}(p_t)$ denoting forward and backward flow from p_t to p_{t+1} and p_{t-1} as additional supervision and formulate $\mathcal{L}_{temp}$ as follows:

$$\begin{aligned}\mathcal{L}_{temp} = ||\overleftarrow{fl}(\hat{p_t}) - \overleftarrow{fl}(\mathtt{N}_{q_t,p_t})||_2 + ||\overleftarrow{fl}(p_t) - \overleftarrow{fl}(\mathtt{N}_{p_t,q_t})||_2 \\ +||\overrightarrow{fl}(\hat{p_t}) - \overrightarrow{fl}(\mathtt{N}_{q_t,p_t})||_2 + ||\overrightarrow{fl}(p_t) - \overrightarrow{fl}(\mathtt{N}_{p_t,q_t})||_2,\end{aligned}$$

where as q_t is the tracked point, we can simply have $\overleftarrow{fl}(q_t) = q_t - q_{t-1}$ and $\overrightarrow{fl}(q_t) = q_t - q_{t+1}$. Please see the Section 5.3.2 for how we estimate $\overleftarrow{fl}(\mathtt{N}_{q_t,p_t})$. The last term $\mathcal{L}_{KL}$ is the KL-divergence loss [34] on the encoding $\boldsymbol{z_t}$ to enforce similarity with a normal distribution $\mathcal{N}(0,1)$.

Hair Volumetric Decoder. In parallel to the point decoder $\mathcal{D}$, we optimize a hair volumetric decoder that regresses a volumetric radiance field around each of the hair points. The hair volumetric primitives $\boldsymbol{V}_t \in \mathbb{R}^{N_{prim} \times 4 \times m^3}$ store RGB and alpha in resolution of m^3. We use a decoder similar to HVH [102] to regress the volume payload. The pose of each volumetric primitive is directly determined by the output of the point decoder: q_t and $\boldsymbol{dir}(q_t)$. We denote $\boldsymbol{R}_t^{p,n} \in SO(3)$ and $\boldsymbol{d}_t^{p,n} \in \mathbb{R}^3$

as the $\boldsymbol{n}$th volume-to-world rotation and translation of hair volume and per-hair-volume scale $s_t^{p,n}$ as the $\boldsymbol{n}$th element in scale $\boldsymbol{s_t}$. Similarly, $\boldsymbol{d}_t^n = q_t^n$ is the $\boldsymbol{n}$th element of q_t. Given the head center $\boldsymbol{x}_t^c$ extracted from the head vertices $\boldsymbol{x}_t$ and hair head direction as $\bar{\boldsymbol{h}}_t^n = q_t^n - \boldsymbol{x}_t^c$, we formulate the rotation $\boldsymbol{R}_t^{p,n}$ as $\boldsymbol{R}_t^{p,n} = [\boldsymbol{l}(q_t^n), \boldsymbol{l}(q_t^n \times \bar{\boldsymbol{h}}_t^n), \boldsymbol{l}(q_t^n \times (q_t^n \times \bar{\boldsymbol{h}}_t^n))]^T$, where $\boldsymbol{l}(x) = x/||x||_2$ is the normalization function. The output of the head model is similar to the hair part except that it is modeling head related (non-hair) regions. We denote the head volume payload as $\boldsymbol{U}_t \in \mathbb{R}^{N_{prim} \times 3 \times m^3}$ and head related rotation $\boldsymbol{R}_t^{x,n} \in SO(3)$, translation $\boldsymbol{d}_t^{x,n} \in \mathbb{R}^3$ and scale $\boldsymbol{s}_t^{x,n} \in \mathbb{R}$.

Differentiable Volumetric Raymarching. Given all volume rotations $\boldsymbol{R}_t^{all} = [\boldsymbol{R}_t^{x,n}, \boldsymbol{R}_t^{p,n}]$, translations $\boldsymbol{d}_t^{all} = [\boldsymbol{d}_t^{x,n}, \boldsymbol{d}_t^{p,n}]$, scales $\boldsymbol{s}_t^{all} = [\boldsymbol{s}_t^{x,n}, \boldsymbol{s}_t^{p,n}]$ and local radiance fields $\boldsymbol{V}_t^{all} = [\boldsymbol{U}_t, \boldsymbol{V}_t]$, we can render them into image $\mathcal{I}_{cam_i}$ and compare it with I_{cam_i} to optimize all volumes. Using an optimized BVH implementation similar to MVP [47], we can efficiently determine how each ray intersects with each volume. We define a ray as $\boldsymbol{r}(\mathfrak{p}, l) = \boldsymbol{o}(\mathfrak{p}) + l\boldsymbol{v}(\mathfrak{p})$ shooting from pixel $\mathfrak{p}$ in direction of $\boldsymbol{v}(\mathfrak{p})$ with a depth l in range of (l_{min}, l_{max}). The differentiable formation of an image given the volumes can then be formulated as below:

$$\mathcal{I}_{\mathfrak{p}} = \int_{l_{min}}^{l_{max}} \mathbf{V}_{t,rgb}^{all}(\mathbf{r}_{\mathfrak{p}}(l)) \frac{dT(l)}{dl} dl,$$
$$T(l) = min(\int_{l_{min}}^{l} \mathbf{V}_{t,\alpha}^{all}(\mathbf{r}_{\mathfrak{p}}(l)) dl, 1),$$

where $\boldsymbol{V}_{t,rgb}^{all}$ is the RGB part of $\boldsymbol{V}_t^{all}$ and $\boldsymbol{V}_{t,\alpha}^{all}$ is the alpha part of $\boldsymbol{V}_t^{all}$. To get the full rendering, we composite the rendered image as $\tilde{\mathcal{I}}_{\mathfrak{p}} = \mathcal{I}_{\mathfrak{p}} + (1 - \mathcal{A}_{\mathfrak{p}}) I_{\mathfrak{p},bg}$ where $\mathcal{A}_{\mathfrak{p}} = T(l_{max})$ and $I_{\mathfrak{p},bg}$ is the background image. We optimize the following loss to train the volume decoder:

$$\mathcal{L}_{pho} = ||\tilde{\mathcal{I}}_{\mathfrak{p}} - I_{\mathfrak{p},gt}||_1 + \omega_{VGG} \mathcal{L}_{VGG}(\tilde{\mathcal{I}}_{\mathfrak{p}}, I_{\mathfrak{p},gt}),$$

where $\mathcal{L}_{VGG}$ is the perceptual loss in [29] and $I_{\mathfrak{p},gt}$ is the ground truth pixel value of $\mathfrak{p}$. We find that the usage of a perceptual loss yields more salient rendering results.

However, as we optimize both $\boldsymbol{U}_t$ and $\boldsymbol{V}_t$ from the images, texture bleeding between the hair volume $\boldsymbol{V}_t$ and the head volume $\boldsymbol{U}_t$ becomes a problem. The texture bleeding issue is especially undesirable when we want to treat the hair and head separately, for example, when we want to animate new hair to a head. To prevent this, we additionally render a hair mask map to regularize both $\boldsymbol{V}_{t,\alpha}$ and $\boldsymbol{U}_{t,\alpha}$. We denote the ground truth hair mask as $M_{\mathfrak{p},gt}$ and the rendered hair mask as

$\mathcal{M}_\mathfrak{p}$:

$$\mathcal{M}_\mathfrak{p} = \int_{l_{min}}^{l_{max}} \mathbf{V}_{t,\mathbb{1}}^{all}(\mathbf{r}_\mathfrak{p}(l)) \frac{dT(l)}{dl} dl,$$
$$T(l) = min(\int_{l_{min}}^{l} \mathbf{V}_{t,\alpha}^{all}(\mathbf{r}_\mathfrak{p}(l)) dl, 1),$$

where $\mathbf{V}_{t,\mathbb{1}}^{all}$ is all one volume if it belongs to $\boldsymbol{V}_t$ otherwise zero. We formulate the segmentation loss as $\mathcal{L}_{mask} = ||\mathcal{M}_\mathfrak{p} - M_{\mathfrak{p},gt}||_1$. The final objective for training the whole autoencoder is $\mathcal{L} = \mathcal{L}_{geo} + \mathcal{L}_{pho} + \omega_{mask}\mathcal{L}_{mask}$.

5.2.2 Dynamic Model

In the second stage, we aim to build a dynamic model that can evolve hair states over time without relying on per-frame hair observation as a driving signal. To achieve that, we leverage the embedding space of hair states built at the state compression stage and train a model that performs the hair state transfer in a supervised manner. To this end, we build a temporal transfer module ($\mathtt{T^2M}$) of hair dynamic priors, that evolves the hair state and can produce hair animation based on the indirect driving signals of head motion and head-relative gravity direction in a self-evolving manner. The design of $\mathtt{T^2M}$ is similar to a hair simulator except that it is fully data-driven. One of the inputs to $\mathtt{T^2M}$ is the encoding $\boldsymbol{z}_{t-1}$ of the previous time step. At the same time, $\mathtt{T^2M}$ is also conditioned on the head per-vertex displacement $\boldsymbol{h}_t = \boldsymbol{x}_t - \boldsymbol{x}_{t-1}$ and $\boldsymbol{h}_{t-1} = \boldsymbol{x}_{t-1} - \boldsymbol{x}_{t-2}$ from the previous two time steps as well as head relative gravity direction $\boldsymbol{g}_t \in \mathbb{R}^3$ at the current time step. $\mathtt{T^2M}$ will then predict the next possible state $\hat{\boldsymbol{z}}_t$ from $\mathtt{T^2M}$ based on those inputs. Similar to the design of the VAE, the output of $\mathtt{T^2M}$ is a distribution instead of a single vector. To be more specific, the mean and standard deviation of $\hat{\boldsymbol{z}}_t$ are $\mu(\hat{\boldsymbol{z}}_t), \delta(\hat{\boldsymbol{z}}_t) = \mathtt{T^2M}(\boldsymbol{z}_{t-1}|\boldsymbol{h}_t, \boldsymbol{h}_{t-1}, \boldsymbol{g}_t)$. During training, we take $\hat{\boldsymbol{z}}_t = \mu(\hat{\boldsymbol{z}}_t) + \mathfrak{n} \odot \delta(\hat{\boldsymbol{z}}_t)$ and during testing $\hat{\boldsymbol{z}}_t = \mu(\hat{\boldsymbol{z}}_t)$. The per point normal distribution vector, $\mathfrak{n}$, is the same shape as $\delta(\hat{\boldsymbol{z}}_t)$ and $\odot$ is the element-wise multiplication.

Training Objectives. We denote the point encoder as $\mathcal{E}(\cdot)$ and the point decoder as $\mathcal{D}(\cdot)$. Across the training of $\mathtt{T^2M}$, we freeze the parameters of both, $\mathcal{E}$ and $\mathcal{D}$. We denote the unordered point cloud at frame t as p_t, its corresponding encoding as $\mu(\boldsymbol{z}_t), \delta(\boldsymbol{z}_t) = \mathcal{E}(p_t)$ and the tracked point cloud as $q_t = \mathcal{D}(\boldsymbol{z}_t)$. The following loss enforces the prediction of $\mathtt{T^2M}$ to be similar to its ground truth:

$$\mathcal{L}_{mse} = ||\mu(\hat{\boldsymbol{z}}_{t+1}) - \mu(\boldsymbol{z}_{t+1})||_2 + ||\delta(\hat{\boldsymbol{z}}_{t+1}) - \delta(\boldsymbol{z}_{t+1})||_2$$
$$\mathcal{L}_{cos} = -\boldsymbol{cos}(\mu\hat{\boldsymbol{z}}_{t+1}, \mu\boldsymbol{z}_{t+1}) - \boldsymbol{cos}(\delta\hat{\boldsymbol{z}}_{t+1}, \delta\boldsymbol{z}_{t+1})$$
$$\mathcal{L}_{ptsmse} = ||\mathcal{D}(\hat{\boldsymbol{z}}_{t+1}) - \mathcal{D}(\boldsymbol{z}_{t+1})||_2,$$

where we not only minimize the ℓ_2 distance between $\hat{\boldsymbol{z}}_{t+1}$ and $\boldsymbol{z}_{t+1}$, but also enforce the cosine similarity and the corresponding tracked point cloud to be equivalent. To adapt $\mathtt{T^2M}$ to the tracked point cloud q_t, we compute the above loss again but using q_t as input, where we generate the corresponding encoding as $\boldsymbol{z}'_t$ from $\mathcal{E}(q_t)$ and its prediction $\hat{\boldsymbol{z}}'_{t+1}$ from $\mathtt{T^2M}(\boldsymbol{z}'_t|\boldsymbol{h}_t, \boldsymbol{h}_{t-1}, \boldsymbol{g}_t)$:

$$\mathcal{L}_{mse,cyc} = ||\mu(\hat{\boldsymbol{z}}'_{t+1}) - \mu(\boldsymbol{z}_{t+1})||_2 + ||\delta(\hat{\boldsymbol{z}}'_{t+1}) - \delta(\boldsymbol{z}_{t+1})||_2$$
$$\mathcal{L}_{cos,cyc} = -\boldsymbol{cos}(\mu\hat{\boldsymbol{z}}'_{t+1}, \mu\boldsymbol{z}_{t+1}) - \boldsymbol{cos}(\delta\hat{\boldsymbol{z}}'_{t+1}, \delta\boldsymbol{z}_{t+1})$$
$$\mathcal{L}_{ptsmse,cyc} = ||\mathcal{D}(\hat{\boldsymbol{z}}'_{t+1}) - \mathcal{D}(\boldsymbol{z}_{t+1})||_2.$$

Similar to how we train our autoencoder, we also enforce two KL divergence losses on both the predicted $\hat{\boldsymbol{z}}_{t+1}$ and $\hat{\boldsymbol{z}}'_{t+1}$ with a normal distribution $\mathcal{N}$. The final objective for training the $\mathtt{T^2M}$ is a weighted sum of the above eight terms.

Animation. Given an initialized hair state, our dynamic model $\mathtt{T^2M}$ can evolve the hair state into future states conditioned on head motion and head-relative gravity direction. One straightforward implementation of the $\mathtt{T^2M}$ would be to directly propagate the hair state encoding $\boldsymbol{z}_t$. However, in practice, we find this leads to severe drift in the semantic space. As a simple feed forward neural network, $\mathtt{T^2M}$ can not guarantee that its output is noise free. The noise in the output becomes even more problematic when we use $\mathtt{T^2M}$ in a recurrent manner, where the output noise will aggregate and lead to drift. To remedy this, instead of propagating the encoding $\boldsymbol{z}_t$ directly, we reproject the predicted encoding $\boldsymbol{z}_t$ by the point autoencoder $\mathcal{E}$ and $\mathcal{D}$ every time for denoising. To be more specific, we acquire the de-noised predicted hair encoding $\hat{\boldsymbol{z}}_{t+1} = \mathcal{E}(\mathcal{D}(\hat{\boldsymbol{z}}_{t+1}))$ from the raw prediction $\hat{\boldsymbol{z}}_{t+1}$ of $\mathtt{T^2M}$. The use of the point autoencoder $\mathcal{E}$ and $\mathcal{D}$ can help us remove the noise in $\boldsymbol{z}_{t+1}$, as the point cloud encoder $\mathcal{E}$ can regress the mean μ_{t+1} and standard deviation δ_{t+1} of $\boldsymbol{z}_{t+1}$ separately by using q_{t+1} as an intermediate variable. Thus, we can extract the noise free part of $\boldsymbol{z}_{t+1}$ by taking the mean μ_{t+1} regressed from $\mathcal{E}$. Please see our experiments for further details of this approach.

5.3 Experiments

In order to test our proposed model, we conduct experiments on both the hair motion data set presented in HVH [102] and our own dataset with longer sequences following a similar capture protocol as HVH [102]. We collect a total of four different hair wig styles with scripted head motions like nodding, swinging and tilting. We also collect an animation test set with the same scripted head motions performed by different actors wearing a wig cap, which we will refer to as "bald head motion sequence". The animation test set contains both single view captures from a smart

phone and multiview captures. The total length of each hair wig capture is around 1-1.5 minutes with a frame rate of 30Hz. A hundred cameras are used during the capture where 93 of them are used to obtain training views and the rest are providing held-out test views. We split each sequence into two folds with similar amounts of frames and train our model exclusively on the training portion of each sequence.

5.3.1 Network Architecture

Here we provide details about how we design our neural networks and further information about training.

Encoder. As training a point cloud encoder solely is extremely unstable, we first train an image encoder and use it as a teacher model to train the point cloud encoder. In practice, we train two encoders for our hair branch together. Here we first illustrate the structure of both encoders. We will go back to how we train them and use them later. One of the encoders is an image encoder which is a convolutional neural network (CNN) that takes multiple view images as input. We denote the image encoder as $\mathcal{E}_{img}$ The other one is $\mathcal{E}$ which is a PointNet encoder that takes either an unordered hair point cloud p_t or a tracked hair point cloud q_t as input. Positional encoding [61] is applied to the raw point cloud coordinate before it is used as the input to the network. We find this is very effective to help the network in capturing high frequency details. In practice, we use frequencies of x^2 where x ranges from 1 to 7. We show the detailed architecture of $\mathcal{E}_{img}$ in Tab 5.1. The architecture of the point cloud encoder $\mathcal{E}$ is shown in Tab 5.2. Both of the two encoders $\mathcal{E}$ and $\mathcal{E}_{img}$ can produce a latent vector in size of 256, which are supposed to describe the same content. Their output will be passed to $\mathcal{E}_\mu$ and $\mathcal{E}_\sigma$ which are two linear layers that produce $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ of $\boldsymbol{z_t}$ respectively.

Point Decoder. We use a 3-layer MLP as the point decoder $\mathcal{D}$, which takes a 1d latent code $\boldsymbol{z_t}$ as input and outputs the coordinate of the corresponding tracked point cloud q_t. We show the architecture of $\mathcal{D}$ in Tab 5.3.

Volume Decoder. The volumetric model is a stack of 2D deconv layers. We align the x-axis and y-axis of each volume and put them onto a 2D imaginary UV-space. Then we convolve on them to regress the z-axis content for each of the x,y position. We show the architecture of the volume decoder in Tab 5.4. In our setting, we have two seperate volume decoder for both RGB volume and alpha volume.

Dynamic Model. We use three different inputs to the dynamic model $\texttt{T}^2\texttt{M}$, namely the hair encoding $\boldsymbol{z}_{t-1}$ at the previous frame $t-1$, the head velocity $\boldsymbol{h}_{t-1}$ and $\boldsymbol{h}_{t-2}$ from the previous two frames $t-1$ and $t-2$, and the head relative gravity direction $\boldsymbol{g}_t$ at the current frame t. We first seperately encode $\{\boldsymbol{h}_{t-1}, \boldsymbol{h}_{t-2}\}$ and $\boldsymbol{g}_t$ into two 1d vectors. Both of them have 128 dimensions. Then, we concatenate them together with encoding $\boldsymbol{z}_{t-1}$ as the input to another MLP to regress the next possible hair

	Encoder $\mathcal{E}_{img}$
1	`Conv2d`(3, 64)
2	`Conv2d`(64, 64)
3	`Conv2d`(64, 128)
4	`Conv2d`(128, 128)
5	`Conv2d`(128, 256)
6	`Conv2d`(256, 256)
7	`Conv2d`(256, 256)
8	`Flatten`()
9	`Linear`(256$\times n_{inpimg}\times$15, 256)

Table 5.1: **Encoder $\mathcal{E}_{img}$ architecture**. Each `Conv2d` layer in the encoder has a kernel size of 3, stride of 1 and padding of 1. Weight normalization [82] and untied bias are applied. After each layer, except for the last two parallel fully-connected layers, a Leaky ReLU [54] activation with a negative slope of 0.2 is applied. Then a downsample layer with a stride of 2 is applied after every conv2d layer. The first linear layer takes the concatenation of all towers from different image views as input. n_{inpimg} stands for much many views we take.

state encoding $\boldsymbol{z}_t$. As in Tab. 5.5, we show the flow of `T`^{2}`M`. For the head velocity branch, we first extract the per-vertex velocity $\boldsymbol{h}_{t-1} = \boldsymbol{x}_t - \boldsymbol{x}_{t-1}$ where $\boldsymbol{x}_t$ is the coordinate of the tracked head mesh at frame t. To be noted, here the $\boldsymbol{h}_{t-1}$ contains only the information of the rigid head motion but not any other non-rigid motion like expression change. This representation of head motion is redundant theoretically, but we find it helps our network to converge better when compared to just using the pure 6-DoF head rotation and translation. We then reshape it and use it as the input to a two layer MLP to extract a 1d encoding of size 128. For the gravity branch, we first encode the gravity direction $\boldsymbol{g}_t$ with cosine encoding [61]. The output of the dynamic model is the mean μ_{t+1} and standard deviation σ_{t+1} of the predicted hair state $\boldsymbol{z}_{t+1}$.

5.3.2 Training details

Dataset and Capture Systems. Following the setting in HVH [102], we also captured several video sequences with scripted hair motion performed under different hair styles for animation tests. During the capture, we ask the participants to put on different kind of hair wigs and perform a variety of head motions like nodding, swinging and tilting. They performed these actions multiple times and at both slow and fast speed. In summary, we collect a dataset of four different hairstyles with

	Encoder $\mathcal{E}$
1	`Conv2d`(3, 128)
2	`Conv2d`(128, 256)
3	`Conv2d`(256, 256)
4	`Conv2d`(256, 256)
5	`Conv2d`(256, 512)
6	`Conv2d`(512, 512)
7	`Conv2d`(512, 512)
8	`Conv2d`(512, 1024)
8	`MAM pooling`()
9	`Linear`(1024$\times$3, 512)
10	`Linear`(512, 256)
11	`Linear`(256, 256)

Table 5.2: **Encoder $\mathcal{E}$ architecture**. We use a $\mathcal{E}$ structure similar to PointNet [74]. All `Conv2d` uses a kernel of 1 and stride of 1, which serves as a shared MLP. We only use `Conv2d` for simpler implementation. After each `Conv2d` layer, a Leaky ReLU [54] activation with a negative slope of 0.2 is applied. Then we use a MAM pool layer to aggregate features from all points. MAM stands for min, avarage and max pooling, which concatenates the results of min, average and max pooling into one. Then, two linear layers are applied to the output of MAM pooling and generate a 256 latent vector.

	Decoder $\mathcal{D}$
1	`Linear`(256, 256)
2	`Linear`(256, 256)
2	`Linear`(256, 4096$\times$3)

Table 5.3: **Decoder $\mathcal{D}$ architecture**. We use an MLP with three `Linear` layers as the decoder $\mathcal{D}$. After each layer except the last layer, a Leaky ReLU [54] activation with a negative slope of 0.2 is applied.

varying appearance and topology and 6 different motions for each hairstyle. The total length for the capture of each hairstyle is around 90 seconds. To collect a demonstration set for animation, we also ask the participants to put on a hair net (bare head) and perform the same set of motions as when they are wearing a hair wig.

Hair Point Flow Estimation. There are three steps for computing the hair point flow, namely per-point feature descriptor extraction, feature matching and flow filtering. In the first step, we compute a per-point feature descriptor based on the

	Volume Decoder	
	global encoding $\boldsymbol{z}_t$	per-point hair feature
	repeat	
	concat	
1	`Linear`(320, 512)	
2	`deconv2d`(512, 256)	
3	`conv2d`(256, 256)	
4	`deconv2d`(256, 256)	
5	`conv2d`(256, 256)	
6	`deconv2d`(256, 128)	
7	`conv2d`(128, 128)	
8	`deconv2d`(128, 16×ch)	

Table 5.4: **Architecture of the Volume Decoder**. We first repeat the global encoding $\boldsymbol{z}_t$ into the shape of the per-point hair feature. The per-point hair feature is a tensor that is shared across all time frames. We then concatenate those two into one. Each layer except for the last one is followed by a Leaky ReLU layer with a negative slope of 0.2. Each `deconv2d` layer has a filter size of 4, stride size of 2 and padding size of 1. Each `conv2d` layer has a filter size of 3, stride size of 1 and padding size of 1. ch stands for the channel size of the output. It is set to 3 if it is an rgb decoder and 1 for a alpha decoder.

	Temporal Transfer Module (`T`^{2}`M`)		
1	head velocity $\{\boldsymbol{h}_{t-1}, \boldsymbol{h}_{t-2}\}$	head relative gravity $\boldsymbol{g}_t$	hair state $\boldsymbol{z}_{t-1}$
2	`Linear`(7306×3, 256)	cosine encoding	
3	`Linear`(256, 128)		
4	`Linear`(539, 256)		
5	`Linear`(256, 256)		
6	`Linear`(256, 256)		
7	`Linear`$(256, 256)$	`Linear`(256, 256)	

Table 5.5: **Temporal Transfer Module** (`T`^{2}`M`). We first encode the head velocity $\{\boldsymbol{h}_{t-1}, \boldsymbol{h}_{t-2}\}$ and head relative gravity $\boldsymbol{g}_t$ into 1d vectors, with a 2-layer MLP and cosine encoding respectively. Then we concatenate hair state $\boldsymbol{z}_{t-1}$ with those vectors to serve as the input to another MLP. The last two layers will be regressing the mean μ_{t+1} and standard deviation σ_{t+1} of the predicted hair state $\boldsymbol{z}_{t+1}$. All `Linear` except for the last two are followed by a Leaky ReLu activation with a negative slope of 0.2.

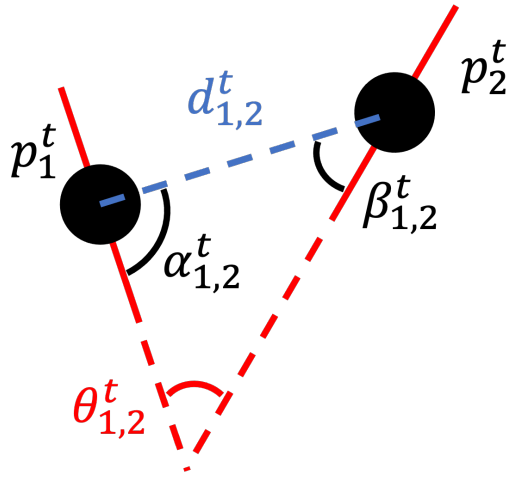

Figure 5.3:

distribution of each point's local neighboring points. In the second step, we match the points from two adjacent time steps based on the similarity between their feature descriptor. In the last step, we filter out outlier flows that are abnormal.

To compute the point feature descriptor, we construct Line Feature Histograms (LFH) inspired by Point Feature Histograms (PFH) [80]. The LFH is a historgram of a 4-tuple that describes the spatial relationship between a certain point $\boldsymbol{p}_1^t$ and its neighboring point $\boldsymbol{p}_2^t$. As shown in Fig. 5.3, we visualize two points $\boldsymbol{p}_1^t \in \mathbb{R}^3$ and $\boldsymbol{p}_2^t \in \mathbb{R}^3$ from the same time step t. Given $\boldsymbol{p}_1^t$ and $\boldsymbol{p}_2^t$, we define the following four properties that describe their spatial relationship. The first one is the relative position of $\boldsymbol{p}_2^t$ with respect to $\boldsymbol{p}_1^t$, which is $\boldsymbol{d}_{1,2}^t = \boldsymbol{p}_2^t - \boldsymbol{p}_1^t$. Then we can compute the relative distance as $||\boldsymbol{d}_{1,2}^t||_2 \in \mathbb{R}$. The second term is the angle $\theta_{1,2}^t$ between $\boldsymbol{dir}(\boldsymbol{p}_1^t)$ and $\boldsymbol{dir}(\boldsymbol{p}_2^t)$, where $\boldsymbol{dir}(\boldsymbol{x})$ is the line direction of $\boldsymbol{x}$ from [63]. The last two terms are the angles $\alpha_{1,2}^t$ and $\beta_{1,2}^t$ between $(\boldsymbol{dir}(\boldsymbol{p}_1^t), \boldsymbol{d}_{1,2}^t)$ and $(\boldsymbol{dir}(\boldsymbol{p}_2^t), \boldsymbol{d}_{1,2}^t)$ respectively. For all intersections, we take the acute angle, which means $\theta_{1,2}^t, \alpha_{1,2}^t$ and $\beta_{1,2}^t$ are in $[0, \pi/2]$. Thus, the 4-tuple we used to create $LFH(\boldsymbol{p}_1^t)$ is $(||\boldsymbol{d}_{1,2}^t||_2, \theta_{1,2}^t, \alpha_{1,2}^t, \beta_{1,2}^t)$ and we normalize the histogram by its l2 norm. The designed LFH has three good properties. As we use the normalized feature, it is density invariant. As $\theta_{1,2}^t, \alpha_{1,2}^t$ and $\beta_{1,2}^t$ are always acute angles, the feature is also rotation and flip invariant, meaning that is unchanged if we flip or rotate $\boldsymbol{dir}(\boldsymbol{p}_1^t)$ the histogram are still the same. This design helps us to get a more robust feature descriptor for matching. We set the resolution for each entry of the 4-tuple to be 4 and it results in a descriptor in size of 256.

In the second step, we compute the correspondence between points from adjacent time frames t and $t + t_\delta$ where $t_\delta \in \{-1, 1\}$. We use the method from Rusu

et al. [79] to compute the correspondence between two point clouds from t and $t + t_\delta$. To further validate the flow we get, we use several heuristics to filter out some obvious outliers. We first discard all the flows that have a large magnitude. As the flow is computed between two adjacent frames, it should not be large. The second heuristic we use to filter the outliers is cycle consistency, where we compute the flow both forward and backward to see if we can map back to the origin. If the mapped back point departs too far away from the origin, we discard that flow.

Training of Encoder. As mentioned before, we train two encoders $\mathcal{E}$ and $\mathcal{E}_{img}$ together. In practice, we find that directly training $\mathcal{E}$ is not very stable and might not lead to convergence. Thus, we learn the two encoders in a teach-student manner, where we use $\mathcal{E}_{img}$ as a teach model to train $\mathcal{E}$. We denote $\boldsymbol{x}_{img,t}$ as the output of $\mathcal{E}_{img}$ and $\boldsymbol{x}_{pt,t}$ as the output of $\mathcal{E}$. Then, we formulate the following MSE loss to enforce the $\mathcal{E}$ to output similarly to $\mathcal{E}_{img}$:

$$\mathcal{L}_{ts} = ||\boldsymbol{x}_{img,t} - \boldsymbol{x}_{pt,t}||_2,$$

where we restraint the gradient from $\mathcal{L}_{ts}$ from back-propagating to $\mathcal{E}_{img}$ while training.

5.3.3 Evaluation of the State Compression Model

We first test our state compression model to evaluate its ability to reconstruct the appearance of hair and head.

Novel View Synthesis. We compare with volumetric methods like NeRF based methods [40, 95] and volumetric primitives based methods [47, 102] on the data set from HVH [102]. In Tab 5.6, we show the reconstruction related metrics MSE, SSIM, PSNR and LPIPS [119] between predicted images and the ground truth image on hold out views. Our method yields a good balance between perceptual loss and reconstruction loss while keeping both of them relatively low. Furthermore, our method achieves a much higher perceptual similarity with ground truth images. In Fig. 5.4 we show that our method can capture high frequency details and even preserve some fly-away hair strands.

We compare our method with MVP [47] on the longer sequences we captured with scripted head motion. Reconstruction related metrics are shown in Tab.5.7. We found that NeRF-based methods can not fit to longer sequences properly. This problem might be due to the large range of motion exhibited in the videos as well as the length of the video. HVH is not applicable because it does not support hair tracking across segmented sequences of different hair motion. Compared to MVP, we achieve better reconstruction accuracy and improved perceptual similarity be-

	seq01				seq02				seq03			
	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓
PFNeRF	51.25	31.16	0.9269	0.3717	103.41	28.15	0.8659	0.5067	76.59	29.50	0.9000	0.2949
NSFF	50.13	31.21	0.9346	0.3672	90.06	28.75	0.8885	0.4728	83.18	29.10	0.8936	0.3292
NRNeRF	56.78	30.78	0.9231	0.3554	132.16	27.13	0.8549	0.5241	79.83	29.33	0.8987	0.3067
MVP	47.54	31.60	0.9476	0.2587	77.23	29.62	0.9088	0.3051	73.78	29.66	0.9224	0.2455
HVH	41.89	32.17	0.9543	0.2019	59.84	30.69	0.9275	0.2353	71.58	29.81	0.9314	0.2021
Ours	40.34	32.28	0.9558	0.1299	56.47	30.94	0.9329	0.1254	73.65	29.69	0.9247	0.1496

Table 5.6: **Novel view synthesis.** We compute MSE↓, PSNR↑, SSIM↑ and LPIPS↓ comparing rendered and ground truth images on hold-out views. First and second best results are highlighted.

	MSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
MVP	66.21	30.36	0.9291	0.2830
Ours	**29.44**	**34.05**	**0.9657**	**0.1109**

Table 5.7: **Novel View Synthesis on Longer Sequence.**

tween the rendered image and ground truth with the hair specific modeling in our design.

Ablation on Hair/Head Disentanglement. We test how well our model handles hair/head disentanglement compared to previous work. As hair and head are exhibiting different dynamic patterns, disentanglement is usually required, especially to achieve independent controllability for both. We compare with HVH, which implicitly separates the hair and head using the dynamic discrepancy between them and optical flow. In our model, we further facilitate the disentanglement by using semantic segmentation. In Tab 5.8, we show the IoU between the rendered silhouette of hair volumes and ground truth hair segmentatation of the different methods. We visualize the difference in Fig. 5.5. Our method generates a more opaque hair texture with less texture bleeding between hair volumes and non-hair volumes. Moreover, our model creates the hair shape in an entirely data-driven fashion, which yields higher fidelity results than the artist prepared hair in HVH [102].

	Seq01	Seq02	Seq03
HVH [102]	0.6685	0.4121	0.3766
Ours	**0.8289**	**0.9243**	**0.8571**

Table 5.8: **IoU**(↑) between rendered hair silhouette and ground truth hair segmentation. Compared to HVH [102], our method achieved a significant improvement over the hair coverage. There two major reasons for the increase: 1) our hair volume texture is more opaque. 2) The coarse level hair geometry better resemble the hair reconstruction.

Figure 5.4: **Novel View Synthesis.** Compared with previous methods, our method captures hair with more details, including fly-away hair strands and creates an overall more accurate hair reconstruction with perceptually better rendering results.

Ablation on $\mathcal{L}_{VGG}$. We examine the synergy between $\mathcal{L}_{VGG}$ and the ℓ_1 loss for improving the rendering quality. As shown in Tab. 5.9, we find that the perceptual loss has positive effects on the reconstruction performance while the improvements are negated when the weight is too large. In Fig. 5.6, we compare the rendered images using different $\mathcal{L}_{VGG}$ weights. The results are more blurry when not using $\mathcal{L}_{VGG}$ and fewer details are reconstructed, such as fly-away strands.

Ablation on Point Flow Supervision. Although with $\mathcal{L}_{cham}$ we can can already

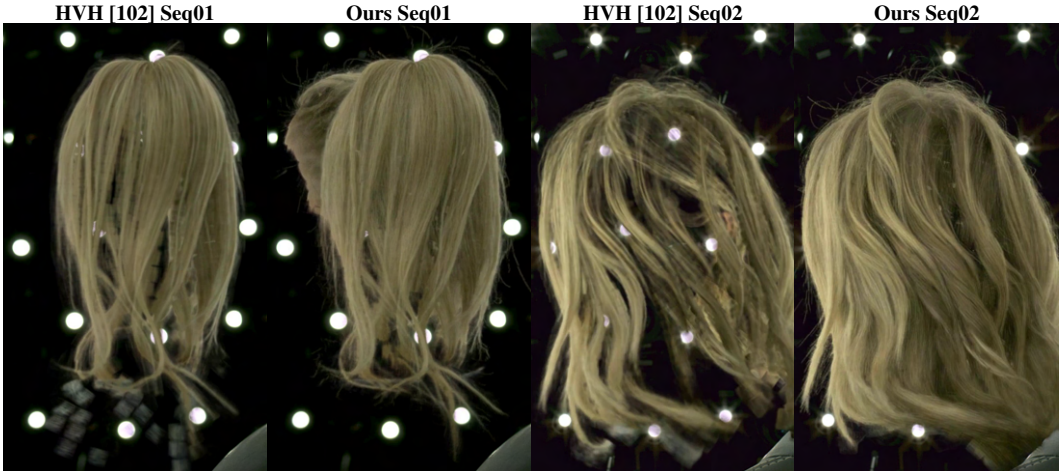

Figure 5.5: **Hair/Head Disentanglement.** By explicitly enforcing the semantic segmentation of head and hair through additional supervision, we learn a more opaque hair texture while the result suffers less from texture bleeding.

	vgg=0.0	vgg=0.1	vgg=0.3	vgg=1.0	vgg=3.0	vgg=10.0
MSE	42.84	42.40	39.94	40.34	40.98	42.82
PSNR	32.04	32.09	32.34	32.28	32.25	32.07
SSIM	0.9544	0.9564	0.9576	0.9558	0.9541	0.9518
LPIPS	0.2021	0.1765	0.1511	0.1299	0.1238	0.1257

Table 5.9: **Ablation on $\mathcal{L}_{VGG}$.** We find using an additional complementary perceptual loss leads to better appearance reconstruction.

optimize a reasonably tracked point cloud p_t, we find that point flow can help remove the jittering in appearance. We show the temporal smoothness enforced by the point flow supervision in Fig. 5.7. Our model learns a more consistent hair texture with less jittering when trained with point flow. Please see the videos in this page[1] for a visualization over time.

Ablation on Different Encoders. We show quantitative evaluations on rendering quality of different encoders on both the SEEN and UNSEEN sequences in Tab. 5.10. Our $\mathcal{E}$ performs similarly to the $\mathcal{E}_{img}$ on the novel views of the SEEN sequence. This result is as expected due to the nature of teach-student model and we train our model on the SEEN sequence with the training views. On the UNSEEN sequence, we find our $\mathcal{E}$ performs better than $\mathcal{E}_{img}$. We hypothesize that this is because there is a smaller domain gap between the point clouds from SEEN sequence

[1] https://ziyanw1.github.io/neuwigs/resources/index.html

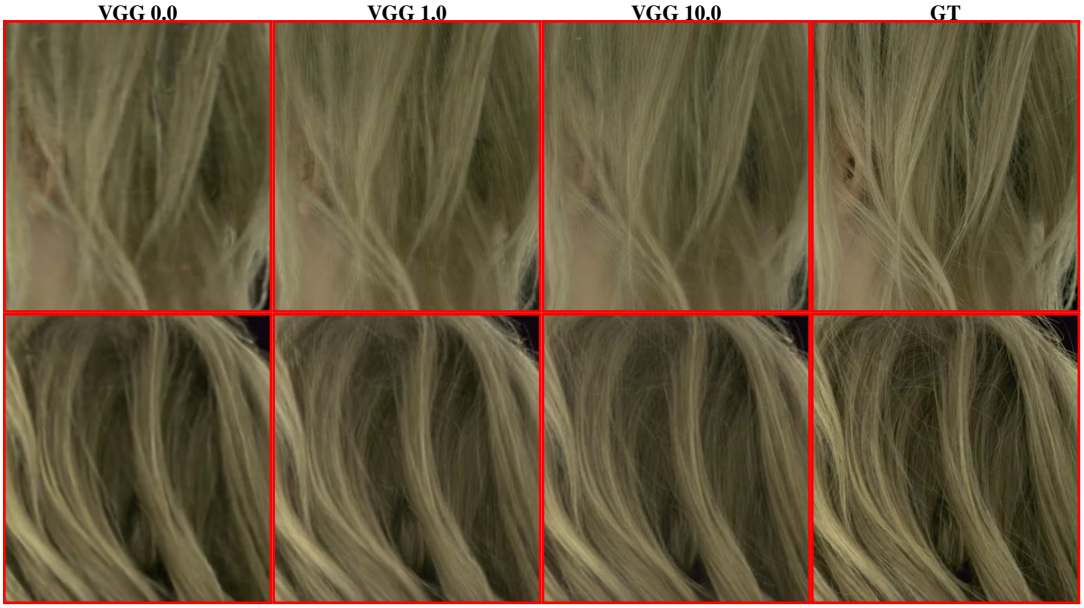

Figure 5.6: **Ablation on $\mathcal{L}_{VGG}$.** Adding a perceptual loss leads to sharper reconstruction results with more salient high frequency textures on parts like single flying away strands and shadows.

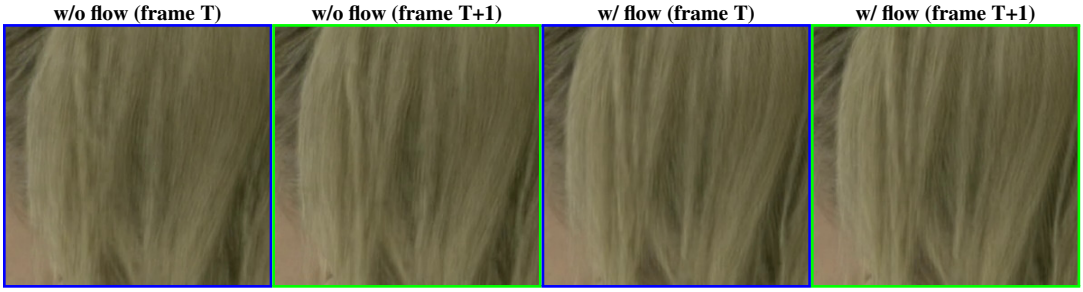

Figure 5.7: **Ablation on Point Flow.** We find that adding point flow to regularize the offsets between temporally adjacent tracked points prevents jittering. Comparison are better visualized here as videos.

and UNSEEN sequence while the multi-view images might vary a lot due to the head motion. The CNN is not good for handling such changes due to the head motion while point encoder can process point clouds with better structure awareness.

5.3.4 Evaluation of the Dynamic Model

Lastly, we perform tests of our animation model. Compared to a per-frame model that takes hair observations as input, the input to our dynamic model and any of

	MSE↓	PSNR↑	SSIM↑	LPIPS↓
$\mathcal{E}$ on SEEN	29.48	34.05	0.9657	0.1109
$\mathcal{E}_{img}$ on SEEN	29.44	34.05	0.9657	0.1109
$\mathcal{E}$ on UNSEEN	34.97	33.21	0.9587	0.1209
$\mathcal{E}_{img}$ on UNSEEN	37.36	32.94	0.9559	0.1333

Table 5.10: **Metrics on Novel Views.** We show quantitative results of different encoders under both SEEN and UNSEEN sequence of the same hair styles.

its variations is a subset of the head motion trajectory and hair point cloud at the initialization frame. As a quantitative evaluation, we compare our model with per-frame driven models using either hair observations or head observations as driving signals. For qualitative evaluation, we render new hair animations for the bald head motion sequences.

Quantitative Test of the Dynamic Model. We evaluate our dynamic model on the test sequences of scripted hair motion capture. The goal is to test whether our dynamic model generates reasonable novel content rather than only testing how well it reconstructs the test sequence. To test the performance of our dynamic model, we treat the model driven by per-frame (**pf**) hair observations as an oracle paradigm to compare with, while our dynamic model does not use any per-frame hair observation as a driving signal. In Tab. 5.12, we compare the rendering quality of our dynamic model with **pf** models and ablate several designs. The best performing dynamic model **(dyn)** has similar performance with the **pf** model even without the per-frame hair observation as driving signal. We find that both adding a cosine similarity loss as an additional objective to MSE and adding a cycle consistency loss helps improve the stability of the dynamic model. Meanwhile, we find adding gravity as an auxiliary input stabilizes our model on slow motions. Presumably because during slow motions of the head, the hair motion is primarily driven by gravity.

We show more detailed comparisons of different dynamic models and per-frame driven models in Tab. 5.11 and Fig. 5.8. We find that our model offers a significant improvement over the per-frame driven model that takes head pose or motion as input. This result is because the hair motion is not only determined by the head pose or the previous history of head pose but also the initial state of the hair. In Fig. 5.8, we visualize how each model drifts by plotting the Chamfer distance between the regressed point cloud and the ground truth point cloud. We find that adding head relative gravity direction can improve the model performance on slow motions.

Ablation for the Point Autoencoder $\mathcal{E} + \mathcal{D}$. Here, we analyze how well the point autoencoder acts as a stabilizer for the dynamic model. We compare two different

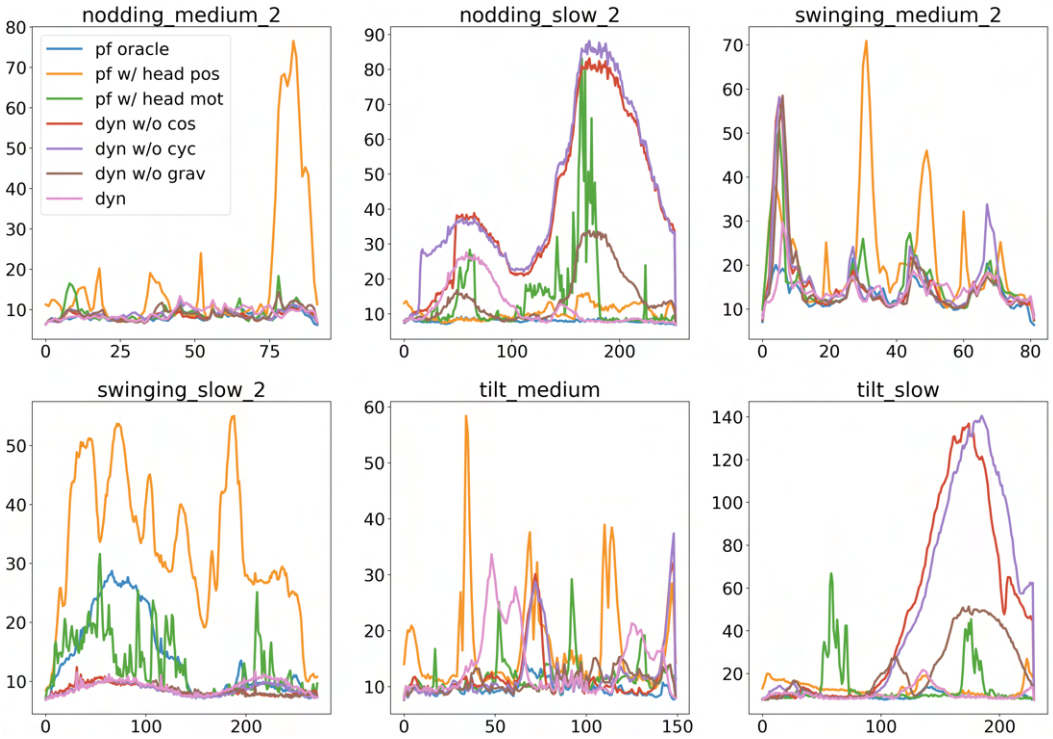

Figure 5.8: **ChamDist vs. time.** We plot Chamfer distance vs. time of different dynamic models to show drifting.

	MSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ChamDis(↓)
pf w/ hair img	37.36	32.94	0.9559	0.1333	10.47
pf w/ hair pts	34.97	33.21	0.9587	0.1209	10.46
pf w/ head pos	47.43	32.01	0.9458	0.1522	18.94
pf w/ head mot	40.25	32.64	0.9508	0.1333	13.31
dyn w/o cos	44.96	32.26	0.9458	0.1327	25.49
dyn w/o cyc	45.22	32.23	0.9453	0.1335	26.79
dyn w/o grav	40.12	32.64	0.9504	0.1268	13.76
dyn	38.49	32.80	0.9532	0.1211	11.12

Table 5.11: **Ablation of Different Dynamic Models.**

models in Fig. 5.9: encprop, that propagates the encoding directly, and ptsprop that propagates the regressed hair point cloud and generates the corresponding encoding from the point encoder. The results are more salient with ptsprop. The improvement of the ptsprop over the encprop is partially because the mapping from point cloud to

	MSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	ChamDis(↓)
pf w/ hair img	**37.36**	**32.94**	**0.9559**	**0.1333**	**10.47**
dyn w/o cos	44.96	32.26	0.9458	0.1327	25.49
dyn w/o cyc	45.22	32.23	0.9453	0.1335	26.79
dyn w/o grav	40.12	32.64	0.9504	0.1268	13.76
dyn	**38.49**	**32.80**	**0.953**2	**0.1211**	**11.12**

Table 5.12: **Ablation of Different Dynamic Models.** We compare different models in terms of rendering quality and tracking accuracy.

encoding is an injective mapping and the point encoder serves as a noise canceller in the encoding space. To further study this behavior, we perform a cycle test on the point encoder, where we add noise n to a certain encoding z and get a noisy version of the encoding $\hat{z}=z+n$ and its corresponding noisy point cloud $\hat{p}$. Then, we predict a cycled encoding $\bar{z}=\mathcal{E}(\mathcal{D}(\hat{z}))$. We compare z, $\hat{z}$ and $\bar{z}$ in Fig. 5.9. z and $\bar{z}$ are consistently close while $\hat{z}$ jitters. This result suggests that the remapping of $\hat{z}$ using $\mathcal{E}$ and $\mathcal{D}$ counteracts the noise n.

Effect of the Initialization. We test how robust our model is to the initialization of the hair point cloud. In Fig. 5.11, we show animation results from models of two different hair styles **(hs)** with different initialization hair point clouds. The results look sharp when the model is matched with the correct hair style, but blurry when we use mismatched hair point clouds for initialization. However, we find our model self-rectifies and returns to a stable state after a certain number of iterations. This self-rectification might be partially due to the model prior stored in the point encoder.

Animation on Bald Head Sequences. We show animation results driven by head motions on in-the-wild phone video captures in Fig 5.1. We find our model generates reasonable motions of hair under natural head motions like swinging or nodding.

We show animation results driven by head motions both on lab multi-view video captures and in-the-wild phone video captures. For results on in-the-wild phone video captures, please refer to the supplemental videos [2]. For phone captures, we ask the participants to face the front camera of the phone and perform different head motions. Then, we apply the face tracking algorithm in [9] to obtain face tracking data that serves as the input to our method. The initial hair state of the phone animation is sampled from the lab captured dataset. We find our model generates reasonable motions of hair under head motions like swinging and nodding.

We also test our model on multi-view video captures from the lab. As shown in

[2]https://ziyanw1.github.io/neuwigs/resources/index.html

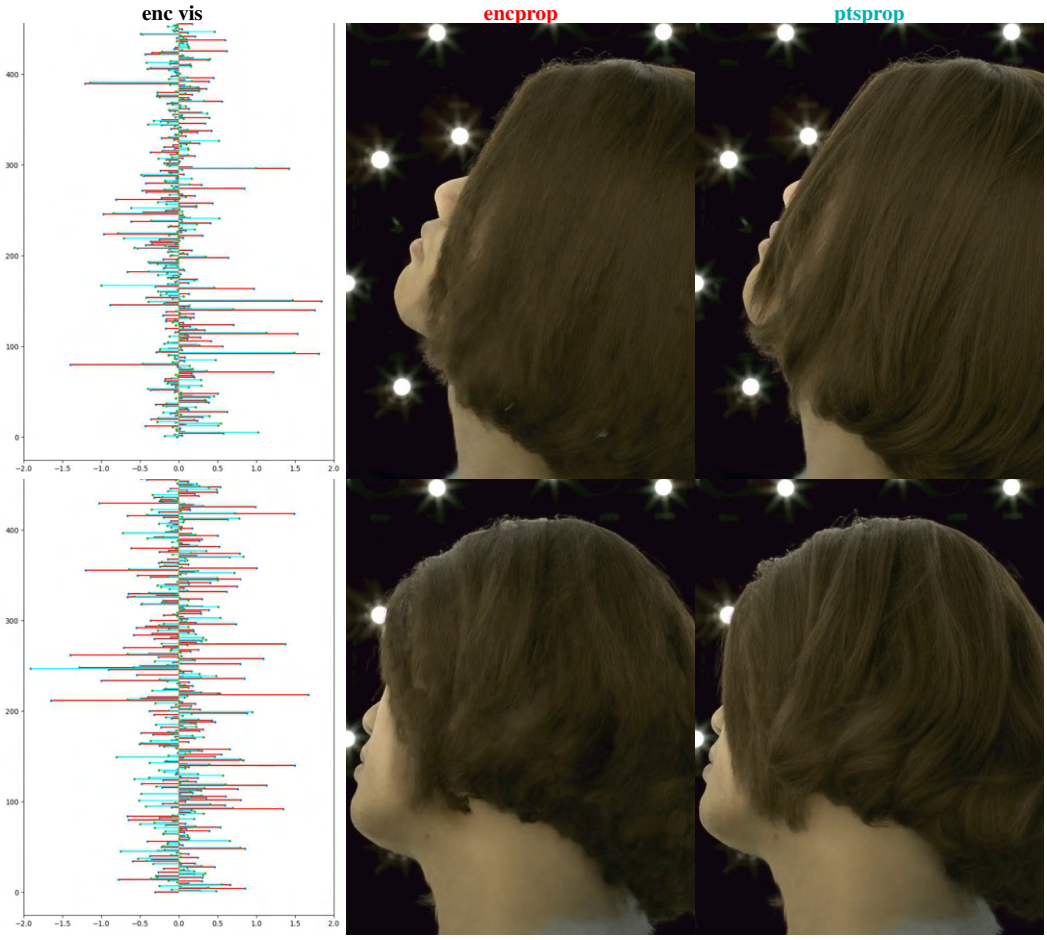

Figure 5.9: **encprop v.s. ptsprop.** ptsprop generates sharper results with less drifting than encprop.

Figs. 5.12 and 5.13, our model generates reasonable hair motions with respect to the head motion while preserving multi-view consistency.

5.4 Discussion

We present a two-stage data-driven pipeline for volumetric hair capture and animation. The core of our method is a 3D volumetric autoencoder that we find useful for both, automatic hair state acquisition and stable hair dynamic generation. The first stage of our pipeline simultaneously performs hair tracking, geometry and appearance reconstruction in a self-supervised manner via an autoencoder-as-a-tracker strategy. The second stage leverages the hair states acquired from the first stage

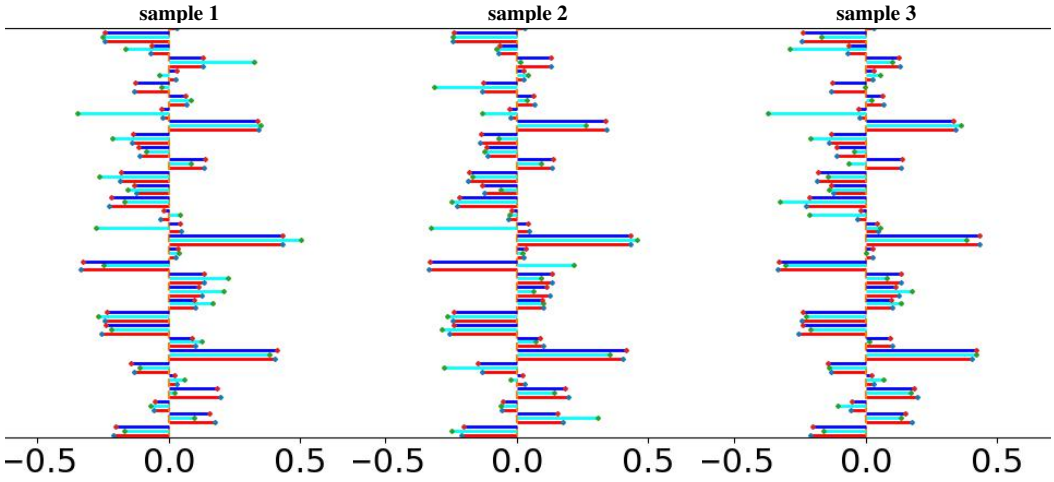

Figure 5.10: **Point Encoder $\mathcal{E}$ as a Stabilizer.** We sample several $\hat{z}$ and corresponding $\bar{z}= \mathcal{E}(\mathcal{D}(\hat{z}))$ with a fixed z and visualize part of them above. As we can see, $\bar{z}$ stays similar to z while $\hat{z}$ jitters.

Figure 5.11: **Effect of Initialization.** We initialize two different models (hs1 mod. and hs2 mod.) with two different hair point clouds (hs1 and hs2) in two time steps. The green box indicates matched initialization while orange indicates mismatched initialization. Although the mismatched initialization shows blurry results at first, the model automatically corrects itself when there is no head motion.

and creates a recurrent hair dynamics model that is robust to moderate drift with the autoencoder as a denoiser. We empirically show that our method performs stable tracking of hair on long and segmented video captures while preserving high fidelity for hair appearance. Our model also supports generating new animations in both, lab- and in-the-wild conditions and does not rely on hair observations.

Limitations. Like many other data-driven methods, our method requires a large amount of diverse training data and might fail with data that is far from the training distribution. Our model is currently not relightable and can not animate new hairstyles. One future direction is to separately model appearance and lighting, which can be learnt from varied lighting captures. Another interesting direction is to learn a morphable hair model for new hairstyle adaptation.

Figure 5.12: **Animation on Bald Sequence.** We animate a straight brown hair with a nodding head.

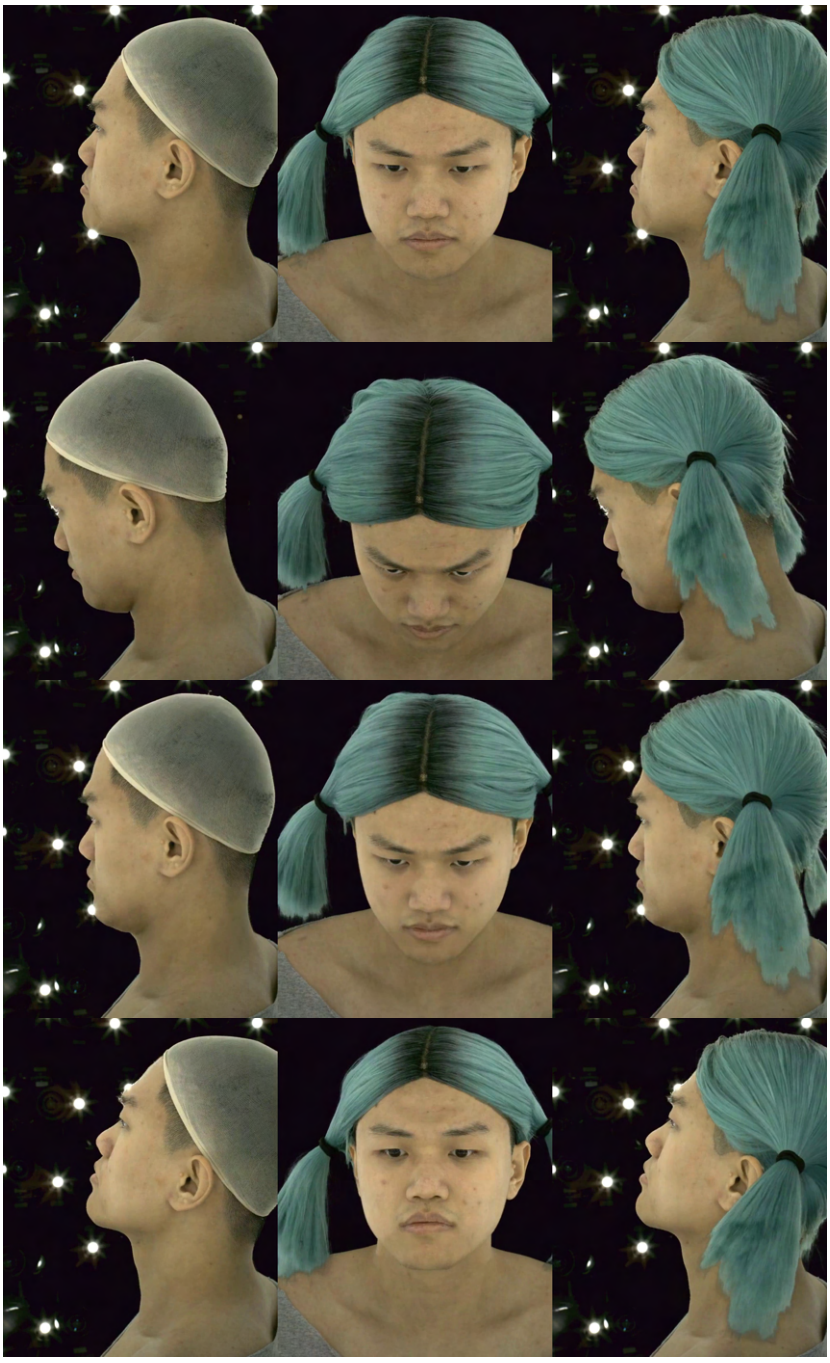

Figure 5.13: **Animation on Bald Sequence.** We animate short blue pigtails with a nodding head.

Figure 5.14: **Animation on Bald Sequence.** We animate curly blonde pigtails with a rotating head.

Figure 5.15: **Animation on Bald Sequence.** We animate burly blonde pigtails with a nodding head.

Figure 5.16: **Animation on Bald Sequence.** We animate a curly ash blonde hair with a nodding head.

Figure 5.17: **Animation on Bald Sequence.** We animate a curly ash blonde hair with a rotating head.

Chapter 6

A Local Appearance Model for Volumetric Capture of Diverse Hairstyles

6.1 Introduction

Creating and capturing high-fidelity 3D human avatars is an essential capability for mixed reality. A high-quality, photorealistic 3D avatar can blur the boundary between the real and virtual world and facilitate many VR/AR applications like social telepresence, virtual gaming and virtual shopping. One critical aspect of achieving a lifelike avatar is accurately capturing and modeling hair, as it plays a vital role in establishing personal identity and achieving personalized avatars. However, hair is potentially challenging to capture due to its complex geometry and high-frequency texture. Furthermore, different hairstyles exhibit large intra-class variance in terms of appearance and shape, which adds to the complexity of efficiently creating personalized avatars for a large group of individuals.

In this work, we look at the problem of how to capture diverse hair appearance for efficient and accurate creation of photorealistic, personalized 3D avatars. Given the complexity of hair geometry, several 3D representations have been explored for modeling a human head avatar in pursuit of modeling accuracy. Mesh-based representations work well for capturing the surface details and are most efficient to store. But they are not well-suited for modeling hairstyles that exhibit volumetric properties. Strand-based representations can capture hair with high accuracy and are easy to manipulate. However, modeling complex hairstyles with strand-based representations as well as rendering them can be computationally expensive. In comparison, many recent volumetric-based methods [19,22,24,30,35,69,70,102,121,122] showcased their success in modeling human avatars with photorealistic hair with diverse geometry and appearance. They can be optimized from images or videos via differentiable raymarching in an end-to-end learning framework. However, the model accuracy comes with the use of a person-specific model which does not generalize and requires an extensive training time. Those properties prohibit them from the efficient creation of personalized avatars for a large group.

To reconcile the efficiency and accuracy in creating personalized avatars for individuals, we present a universal 3D hair appearance model that captures diverse hairstyles with high fidelity and helps to generate a decent hair appearance for personalized avatar creation. Our universal 3D hair appearance model is conditioned on a group of hair feature volumes that are diffused from a 3D hair point cloud with color. Those hair feature volumes are anchored by volumetric primitives that tightly bound the hair point cloud, and reflect the local structures and appearance at a primitive level. To amplify the sparse point cloud for dense appearance modeling, we learn a UNet to transfer those hair feature volumes into dense radiance fields. By spatially compositing those volumetric primitives, we get a set of volumetric radiance fields that fully cover different parts of the hair. Those volumetric radiance fields are not restricted to certain topologies and are flexible with modeling differ-

Figure 6.1: **Pipeline of Our Method.** We present a pipeline to achieve large scale capture of diverse hairstyles for avatar creation. The core of our pipeline is a local UNet that can generate local appearance field conditioned on colored point cloud $\boldsymbol{q}$. Our method is robust to various challenging hairstyles and can generate photorealistic appearance of those hairstyles.

ent hairstyles. The design of learning an appearance prior at a primitive level rather than the hairstyle level is based on the observation that different hairstyles share a more similar pattern within a local region than at a global scale. As hairstyles vary a lot both in geometry and appearance, learning a compact embedding space at a global scale to express such variety is nearly implausible, with a limited amount of training data. Learning a local prior model helps us to achieve data augmentation as it splits each hairstyle into multiple local volumetric primitives.

We conduct extensive experiments on multiview captures of multiple identities as well as in-the-wild captures of new identities under a sparse viewpoints setup. We empirically show that our method outperforms previous state-of-the-art methods [9, 58] in capturing diverse hairstyles with improved quality and generalizing to new hairstyles for personalized avatar creation. Given sparse views as input, we also find that our local prior model serves as a good initialization for more efficient acquisition of personalized avatars with less finetuning. In summary, our contributions are:

- We present a novel volumetric feature representation based on a point cloud with color and a local appearance model that is generalizable to various complex hairstyles.

- We empirically show that our method outperforms previous state-of-the-art approaches in capturing high-fidelity avatars with diverse hairstyles and generating photorealistic appearances for novel identities with challenging hairstyles. Our method also enables the efficient capture of personalized avatars from an iPhone scan.

6.2 Method

Our goal is to achieve efficient and accurate hair appearance capture of a large group with a single model that is also generalizable. We use a compositional volumetric representation for hair modeling. There are several benefits of using such a representation for hair modeling. First, volumetric representation is flexible with complex hair configuration while also yielding a decent rendering quality with high fidelity on detailed geometry. Second, rendering a compositional volumetric representation is more efficient than a volumetric representation, as the raymarching process is guided by a sparse structure to skip empty spaces. To accommodate the diverse topology as well as capture the generalizable prior of various hairstyles, we learn a local hair appearance prior model Ψ_α and Ψ_{rgb} that can regress the compositional volumetric presentation based on sparse color point clouds $\boldsymbol{q}$. The model takes input as a group of local hair feature volumes diffused by the colored hair point cloud, which is agnostic to the ordering of the point cloud and outputs a compositional volumetric representation $\mathbf{V}_{rgb} + \mathbf{V}_\alpha$ that can be rendered from different viewpoints.

6.2.1 Preliminaries: Volumetric Rendering

We render and optimize our compositional volumetric representation with the differentiable volumetric raymarching algorithm in MVP [47]. Given the camera center as $\boldsymbol{c}$ and a ray direction $\boldsymbol{v}(p)$ associated with pixel p, we can define a ray function $\boldsymbol{r}$ as follow:

$$\boldsymbol{r}(p) = \boldsymbol{c} + \tau \boldsymbol{v}(p),$$

where τ is the traversal depth along the ray and $\boldsymbol{v}(p)$ is the direction starting from camera center $\boldsymbol{c}$ and pointing to pixel p.

To render the compositional volumetric representation, we aggregate all the alpha, the RGB values and the semantic labels from the volumetric field. The above image formation process can be formulated as:

$$\begin{aligned}
\mathcal{I}_p &= \int_{\tau_{min}}^{\tau_{max}} \mathbf{V}_{rgb}(\mathbf{r}_p(\tau)) \frac{dT(\tau)}{d\tau} d\tau, \\
\mathcal{M}_p &= \int_{\tau_{min}}^{\tau_{max}} \mathbf{V}_{label}(\mathbf{r}_p(\tau)) \frac{dT(\tau)}{d\tau} d\tau, \\
T(l) &= min(\int_{\tau_{min}}^{\tau_{max}} \mathbf{V}_\alpha(\mathbf{r}_p(\tau)) d\tau, 1),
\end{aligned}$$

where we composite the RGB and semantic label from near to far in a weighted sum manner. $\mathbf{V}_{\clubsuit}(\cdot)$ indicates the volumetric field function that outputs the function value of a specific spatial point, where $\clubsuit$ can be alpha, RGB, or semantic label. For efficient rendering of $\mathbf{V}_{\clubsuit}$, we use a BVH to speed up the process of finding intersections between rays and volumetric primitives following MVP [47].

To get the full rendering, we composite the rendered image as $\tilde{\mathcal{I}}_p = \mathcal{I}_{\mathfrak{p}} + (1 - \mathcal{A}_p)I_{p,bg}$ where $\mathcal{A}_p = T(l_{max})$ and $I_{p,bg}$ is the background image. Similar to NeuWigs [101], we model the hair and head of an avatar in separate layers and render their segmentation map as $\mathcal{M}_p$. Please refer to the supplemental materials for implementation details.

6.2.2 Local Hair Appearance Prior Model

In this part, we describe how we achieve the conditional generation of $\mathbf{V}_{alpha}$ and $\mathbf{V}_{rgb}$ based on sparse colored hair point cloud $\boldsymbol{q}$ with a local appearance prior model. We will first describe how we create the input to the local appearance model with a colored hair point cloud. Then we will introduce our design for the architecture of the local appearance prior model as well as objectives for training the model.

Hair feature volumes. As different hairstyles might be in different topologies and different sizes, it is not practical to learn a model that has a fixed output size to regress the appearance field of both long hair and short hair in the 3D space directly. To achieve modeling across different hairstyles and capture the common appearance prior among them, we learn a local appearance prior model to regress the radiance field for each of those hair volumes in separate runs. Given a colored hair point cloud $\boldsymbol{q}$ defined in a head-centered coordinate system, we prepare the input to the local appearance prior model by first partitioning it into a group of hair feature volumes. Specifically, we first perform furthest point sampling [74] on $\boldsymbol{q}$ to get k centroids $\{\rho_i | i = 1, 2, \cdots, k\}$ that roughly span over the point cloud manifold uniformly. Then for each centroid ρ_i, we diffuse the points into a volume grid Ω^{ρ_i} centered at ρ_i. The volume grid is axis-aligned with the head-centered coordinate system with a length of δ and a grid resolution of m. Each vertex in the grid Ω^{ρ_i} stores both the spatial occupancy and RGB values, where the occupancy is 0 if no point is found within the radius of $\sqrt{3}\delta/2m$ around that vertex otherwise 1. The RGB value will be the mean of the colors from all found points and $\mathbf{0}$ if no point is found. In addition to the occupancy and RGB value, we aggregate the spatial coordinate of each vertex in the head-centered coordinates as well as the per-vertex viewing direction into volume grids Γ^{ρ_i} and Λ^{ρ_i} respectively. Given viewing direction camera center $\boldsymbol{c}$ under the head centered coordinate, we calculate $\Lambda^{\rho_i} = norm(\Gamma^{\rho_i} - \boldsymbol{c})$ where we take the normalized vector between each point in Λ^{ρ_i} and $\boldsymbol{c}$ as the per-vertex viewing direction.

Local appearance UNet. We learn two separate UNets [77] with skip connections that takes volume grid Ω_i^ρ as input and outputs the corresponding dense radiance field $\nu_\alpha^{\rho_i}$ and $\nu_{rgb}^{\rho_i}$ respectively as follow:

$$\nu_\alpha^{\rho_i} = \Psi_\alpha(\Omega_i^\rho, \Gamma^{\rho_i}|\boldsymbol{\theta}_\alpha)$$
$$\nu_{rgb}^{\rho_i} = \Psi_{rgb}(\Omega_i^\rho, \Gamma^{\rho_i}, \hat{\boldsymbol{c}}, \Lambda^{\rho_i}|\boldsymbol{\theta}_{rgb}),$$

where $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_{rgb}$ are the learnable parameters for each networks. By spatially compositing the volumetric primitives $\{\nu_\alpha^{\rho_i}, \nu_{rgb}^{\rho_i}|i = 1, 2, \cdots, k\}$ with respect to their centroids $\{\rho_i|i = 1, 2, \cdots, k\}$, we get $\mathbf{V}_{alpha}$ and $\mathbf{V}_{rgb}$. The UNet $\Psi_\alpha(\cdot)$ that regresses the alpha field of hair takes the occupancy and RGB field Ω^{ρ_i} as well as the grid coordinate Γ^{ρ_i} as input. To model the view conditioned appearance of hair, we learn a separate UNet $\Psi_{rgb}(\cdot)$ to regress the RGB field $\nu_{rgb}^{\rho_i}$ that takes additional input of the per-vertex viewing direction Λ^{ρ_i} as well as the normalized camera center $\hat{\boldsymbol{c}}$ as viewing direction. Λ^{ρ_i} is served as additional information to the input of the UNet $\Psi_{rgb}(\cdot)$ and $\hat{\boldsymbol{c}}$ is injected at the bottleneck level where it is repeated and appended to every hair features at the coarsest resolution map. We find that the usage of Γ^{ρ_i} and Λ^{ρ_i} improves the model's convergence by a large margin, which will be discussed in detail in the experiment. The encoder part of each UNet consists of convolutional layers with a kernel size of 3×3 with stride 1 and 2 to extract the features of Ω_i^ρ at different scales. The decoder part of each UNet consists of convolutional layers with a kernel size of 3×3 with stride 1 and deconvolutional layers with a kernel size of 4×4 with stride 2. At each scale, we use a 1×1 convolutional layer as skip connections to add early conditions from the encoder features to the decoder features respectively. All layers are followed with a LeakyReLU layer as activation. We find that the usage of skip connections can greatly help the network to capture detailed geometry and more salient textures on the regressed hair radiance field.

Training objectives and details. To learn the parameters $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_{rgb}$ of the local appearance model $\Psi_\alpha(\cdot)$ and $\Psi_{rgb}(\cdot)$, we construct image level reconstruction losses. We formulate the training objective $\mathcal{L}$ as below:

$$\mathcal{L} = \mathcal{L}_1 + \lambda_{VGG}\mathcal{L}_{VGG} + \lambda_{seg}\mathcal{L}_{seg},$$

where λ_{VGG} and λ_{seg} are positive values for rebalancing each term in the training objectives. The first term $\mathcal{L}_1$ measures the difference between the rendered image $\tilde{\mathcal{I}}$ and the ground truth image I_{gt}. The second term is a perceptual loss between the rendered image $\tilde{\mathcal{I}}$ and the ground truth image I_{gt}, which aims at enhancing the visual quality and adding high frequency details of the rendered image. The third term is a segmentation loss for better disentangling hair and non-hair region. Please refer to the supplimental materials for more training related details.

TRAIN	MSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
KeypointNeRF [58]	257.42	24.57	0.86	0.3140
Cao *et al.* [9]	159.37	27.12	0.7961	0.3117
Ours	**130.07**	**27.73**	**0.8922**	**0.1993**
TEST	MSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
KeypointNeRF [58]	303.60	23.77	0.8596	0.3389
Cao *et al.* [9]	334.68	23.89	0.7883	0.3511
Ours	**236.46**	**25.08**	**0.8741**	**0.2610**

Table 6.1: **Novel View Synthesis.** We show qualitative results on novel view synthesis. The upper part and lower part of the table report the MSE, PSNR, SSIM and LPIPS computed on the holdout views of training identities and the test identities respectively. Our method achieves a better result on MSE, PSNR and SSIM compared to previous methods [9, 58]. Our method is capable of generating sharp appearance on detailed geometries which leads to improvement on LPIPS by a large margin.

6.3 Experiments

Dataset. We collect a multiview RGB image dataset of multiple identities with diverse personalized hairstyles. The dataset contains lightstage capture of around 260 identities and each capture has around 160 views covering most perspectives around the participant with a focus on the head and hair region. We exclude captures of 8 identities from training our model and use them just for test purposes. For all 256 training identities, we also hold out 7 views for testing and the rest of the views will be used for training.

6.3.1 Novel View Synthesis

We test our model on the task of novel view synthesis and compare it with previous state-of-the-art approaches on generating personalized avatars like the universal prior model(UPM) in Cao *et al.* [9] and a generalizable NeRF model KeypointNeRF [58]. We evaluate different methods using image reconstruction and similarity metrics like MSE, PSRN, SSIM and LPIPS between the ground true image and the reconstructed ones, which are reported in Tab. 6.1. In the upper part of the table, we report the metrics computed on the holdout views of the training identities. This

is supposed to reflect how well each model reconstructs the hair appearance and shape on those training identities. Compared to KeypointNeRF [58], we achieve a lower distortion in terms of reconstructing different hairstyles. When compared with the UPM model [9], we find that our model enjoys a much larger improvement on LPIPS compared to the other reconstruction metrics like MSE, PSNR and SSIM. One of the reasons behind this is that the perceptual metric is more sensitive to high-frequency information as well as the fine-level details in one's appearance. And the UPM model is capable of reconstructing a coarse-level geometry and appearance but fails to capture the fine-level details which our model does a better job on. In Fig. 6.2, the improvements of our methods can be better justified visually, where we show the rendering results of each method on the holdout views of several training identities. On the lower part of Tab. 6.1, we report the metrics computed on the same set of holdout views but of test identities. Fig. 6.3 shows the rendering results of some of those views. We can see that the UPM model [9] and ours both achieve better generalization on the face than KeypointNeRF [58] as a result of awareness of face geometry. Our method can achieve a more detailed hair appearance given sparse inputs like point clouds on never-seen-before identities.

Ablation on input features Ω^{ρ_i}, Γ^{ρ_i} and Λ^{ρ_i}. We ablate on the usage of different input features to the local appearance networks $\Psi_\alpha(\cdot)$ and $\Psi_{rgb}(\cdot)$. Tab. 6.2 shows the performance of our model under different input configurations. The base model is $\Psi_\alpha(\Omega^{\rho_i}) + \Psi_{rgb}(\Omega^{\rho_i}, \hat{\boldsymbol{c}})$, which only takes Ω^{ρ_i} and $\hat{\boldsymbol{c}}$ as input and do not have untied bias for each layer. $+\Gamma^\rho$ represents the model that use Γ^ρ as additional input to both $\Psi_\alpha(\cdot)$ and $\Psi_{rgb}(\cdot)$. $+ub$ stands for adding untied bias to each learnable layer in $\Psi_\alpha(\cdot)$ and $\Psi_{rgb}(\cdot)$.

According to Tab. 6.2, the inclusion of Γ^ρ helps the base model to converge better with improved image reconstruction and perceptual metrics on both training and testing sets. However, adding untied bias solely gives worse results on both data splits. We argue that using Ω^ρ as the only input makes the network to be aware of only local region and to be agnostic of positional information, which further exposes the network to the noise in Ω^ρ. And including additional network parameters like the untied bias in such a setting will make the network even harder to learn. If we combine both Γ^ρ and the untied bias, we get much better performance on the training set. This improvement suggests that using Γ^ρ is more effective than just adding more network parameters, which helps the network to capture the underlying correlation between human hair's appearance and spatial position. Finally, we find that having per-vertex view conditioning Λ^ρ as additional input help the model to achieve the best on both the training and testing set. As different hair geometry leads to different shadow and reflectance patterns, the per-vertex viewing condition Λ^ρ serves as a more informative term than the viewing direction $\hat{\boldsymbol{c}}$ to infer the view-conditioned appearance.

	TRAIN				TEST			
	MSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	MSE(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
base	150.69	27.11	0.8090	0.2039	242.56	25.01	0.8023	0.2556
base+Γ^{ρ}	140.38	27.42	0.8123	0.1973	228.41	25.19	0.8059	0.2479
base+ub	164.04	26.70	0.8022	0.2234	253.61	24.69	0.7969	0.2705
base+Γ^{ρ}+ub	134.60	27.54	0.8108	0.2042	236.46	25.08	0.8081	0.2610
base+Γ^{ρ}+ub+Λ^{ρ}	130.07	27.73	0.8922	0.1993	220.00	25.45	0.8741	0.2472

Table 6.2: **Ablation on different inputs Ω^{ρ_i}, Γ^{ρ_i} and Λ^{ρ_i}.** We evaluate models with different input configurations and report their MSE, PSNR, SSIM and LPIPS on the holdout views of both training and testing data. As we can see, both Γ^{ρ_i} and Λ^{ρ_i} serve as a more effective way to improve the model performance compared to just increasing the model's capacity like with untied bias ub. We also find that the inclusion of per-vertex viewing direction Λ^{ρ_i} improves the model's performance on novel view synthesis by a large margin. We use gold, silver and bronze to indicate first, second and third places.

Ablation on finetuning efficiency. Even though our model generalizes reasonably to unseen identities and creates photorealistic avatars for them, finetuning or online optimization is still needed for getting metrically correct personalized avatars. Thus, we evaluate our model in terms of how well it can help get personalized avatars for novel identities efficiently. Given RGB-D scans of new subjects with novel hairstyles, we finetune our model based on them to capture the personalized 3D avatar. We ablate on both the number of views needed to do finetuning as well as the number of iterations we update our model.

In Fig 6.5, we show how MSE, PSNR, SSIM and LPIPS on test views change across optimization steps. Each curve in Fig 6.5 represents a model with a different finetuning configuration. ft stands for finetuning a pre-trained model on the new identity while noft stands for training the same model on the new identity from scratch. $x$vs indicates that we use a total number of x views to perform the finetuning. For example, 10vs means that we use 10 views for finetuning. To make sure that the finetuning views are not biased to certain viewpoints, we sample the finetuning views from all training views using the furthest point sampling. As we can see, the pre-trained model could give the finetuning a warm start and leads to better convergence under a short amount of training time, which is also robust to the density of finetuning views.

In Fig 6.4, we show the rendering results of a test view under different finetuning/training configurations as in Fig. 6.5. As we can see, at iteration 0, our model can already reconstruct most details about the new hairstyle and after 100 iterations models trained with different numbers of views gets sharp results, while models

trained from scratch have not yet converged.

6.3.2 Personalized Avatar from an iPhone Scan

We demonstrate an application enabled by our method, which is the efficient generation of personalized avatars with an iPhone scan. We have been using a multi-view camera system to collect the training and validation data for learning the local appearance prior model Ψ_α and Ψ_{rgb}. However, a multi-view capture system is expensive to set up and not readily available to create personalized avatars for individuals. To accommodate the need for simplicity and scalability, we seek to use a single RGB-D camera (like an iPhone) to substitute for the multi-view capture system, which might yield a decayed quality of data. We demonstrate that our learned model Ψ_α and Ψ_{rgb} can infill the gaps between in-the-wild capture and a lab-level capture in many ways like accommodating a sparse view and noisy camera tracking setup.

Reconstructing from iPhone Scan. We perform further experiments on reconstructing a personalized avatar from an iPhone scan in sparse views. The data we used is RGB-D data from the iPhone scan. We asked the participant to sit tight and start our iPhone scan by facing the iPhone towards the participant at a distance of 30-50cm and capturing different perspectives of the participant's face, which results in the participant occupying $70 - 80\%$ of the whole image. The resolution of the captured image is 1024×667 and the resolution of the captured depth image is 640×480. To get the iPhone tracking information, we perform ICP tracking using the per-frame depth as well as color information from the RGB-D scan under different perspectives.

We first test our model on in-the-wild avatar creation using the point cloud from the iPhone scan. However, we observe that the quality of the reconstructed avatar is not comparable to those reconstructed from the lightstage capture system. We argue there are two reasons for the decrease in quality. The major one is the lack of explicit modeling of lighting in our appearance model. As the model is trained under the lighting condition from a lightstage capture, it is biased to that lighting distribution and does not generalize well to the in-the-wild lighting condition. Another minor reason lies in the noise in the input to the model. Given the nature of the in-the-wild capture setup, there will be noise in both the depth scan as well as the camera tracking. However, we demonstrate that our method is not severely affected by those noises after a moderate level of online finetuning.

We compare our method with instant-ngp [62] based on the implementation [89] and perform personalized avatar acquisition from the RGB-D scan. We optimize both instant-ngp and our method using five views with training objectives described in Sec 6.2.2. To prepare the input to our method, we fuse the RGB-D scans into a

point cloud given the ICP tracking results. We generate the foreground mask using RVM [41] and masked the background in the iPhone captures to be black with the mask. Both models are optimized for 5min and converge. As shown in Fig. 6.6, our method and instant-ngp create good-quality rendering on the training views. However, we also find that instant-ngp yields floating artifacts under the sparse view training setting. When we render from a none training camera distribution, the quality of instant-ngp decays dramatically. This result suggests that our method is more robust to sparse views training and camera tracking noise with fewer floating artifacts and suffers from less overfitting on training views.

6.4 Conclusion

In this chapter, we develop a method based on compositional volumetric representation for efficient and accurate capturing of human avatars with diverse hairstyles. Towards that goal, we build a universal hair appearance prior model for modeling the appearance of diverse hairstyles. To accommodate the large intra-class variance in hair appearance, we split hairstyles into small volumetric primitives and learn a local appearance model that captures the universal appearance prior at that scale. We empirically show that our model is capable of generating a dense radiance field for a large spectrum of hairstyles with photorealistic appearance and outperforms previous state-of-the-art approaches on both capturing fidelity and generalization. As a result, our method supports applications like generating personalized avatars from in-the-wild scans using sparse views.

6.5 Implementation Details on Volumetric Rendering

Differentiable Volumetric Raymarching in MVP [47]. Following the formulation in Sec 6.2.1, we first explain the implementation details regarding how we aggregate the spatial radiance functions $\mathbf{V}_{rgb}$, $\mathbf{V}_{\alpha}$ and $\mathbf{V}_{label}$ into the renderings as $\mathcal{I}_p$, $\mathcal{M}_p$ and $T(l)$. We use Riemann sum to approximate the integral in Sec 6.2.1. For simplicity, we use a toy example shown in Figure 6.7 for easier illustration. We denote the red dot $\boldsymbol{c}$ as the camera center and the arrow $\boldsymbol{v}(p)$ as the raymarching direction of pixel p. We march the ray with uniform step size which results in sample points (blue dots) along the ray with depth τ_i. And we denote the volumetric primitives that intersect with those sample points as ν_*^j where $*$ can be α, rgb or $label$ and j is the index of the corresponding volumetric primitive. The green ones are the primitives for the non-hair region and the orange ones are the primitives for

the hair region. We use the term $\nu_*^j(\tau_i)$ as the value we sampled from ν_*^j at point τ_i. To get $\nu_*^j(\tau_i)$, we use trilinear interpolation between the nearest vertices of τ_i in ν_*^j. As in MVP, the aggregated α value up to τ_i along the ray is computed as below

$$T(\tau_i) = min(1, \sum_{j=1}^{i} \sum_{k \in j_k} \nu_\alpha^{j\_k}),$$

where j_k is the set for all the indices of the intersected primitives at τ_j. Supposing that all the α values adds up to a value greater than 1 at $\nu_\alpha^{4\_1}$, we will have

$$\begin{aligned}
T(\tau_1) &= \nu_\alpha^{1_1} \\
T(\tau_2) &= T(\tau_1) + \nu_\alpha^{2_1} + \nu_\alpha^{2_2} \\
T(\tau_3) &= T(\tau_2) + \nu_\alpha^{3_1} \\
T(\tau_4) &= 1 \\
T(\tau_5) &= 1,
\end{aligned}$$

where $T(\tau_4)$ and $T(\tau_5)$ will be constant. Thus, we will have the aggregated rgb and $label$ values as

$$\begin{aligned}
\mathcal{I}_p = &\nu_\alpha^{1_1}\nu_{rgb}^{1_1} + \nu_\alpha^{2_1}\nu_{rgb}^{2_1} + \nu_\alpha^{2_2}\nu_{rgb}^{2_2} \\
&+ \nu_\alpha^{3_1}\nu_{rgb}^{3_1} + (1 - T(\tau_3))\nu_{rgb}^{4_1},
\end{aligned}$$

where the above equation will yield 0 gradient with respect to $\nu_*^{4\_2}$, $\nu_*^{4\_3}$ and $\nu_*^{5\_1}$. As in NeuWigs [101], the aggregated label value will be zero in this case as $T(\cdot)$ already saturates at $\nu_\alpha^{4\_1}$. The early termination of raymarching in this case is especially problematic if pixel p is on the hair region. In this case, the label loss will not backpropagate any useful gradients to update ν_α^i to the correct value. To fix that problem, we make the saturation point to be better awared of the other intersecting boxes with a new soft blending formulation. Instead of taking the rgb and label value of the very first box τ_4 intersects, we compute the rgb and label at that point as

$$\begin{aligned}
\mathcal{I}_p = &\nu_\alpha^{1_1}\nu_{rgb}^{1_1} + \nu_\alpha^{2_1}\nu_{rgb}^{2_1} \\
&+ \nu_\alpha^{2_2}\nu_{rgb}^{2_2} + \nu_\alpha^{3_1}\nu_{rgb}^{3_1} \\
&+ (1 - T(\tau_3))\frac{\sum_{i=1}^{3} \nu_\alpha^{4_i}\nu_{rgb}^{4_i}}{\sum_{i=1}^{3} \nu_\alpha^{4_i}} \\
\mathcal{M}_p = &(1 - T(\tau_3))\frac{\sum_{i=2}^{3} \nu_\alpha^{4_i}}{\sum_{i=1}^{3} \nu_\alpha^{4_i}}.
\end{aligned}$$

With a soft blending reformulation, the rendering terms $\mathcal{I}_p$ and $\mathcal{M}_p$ are both aware of all the primitives intersected at the saturation point and the rendering formulation is no longer affected by the stochasticity in the primitive sorting at the same point.

6.6 Training details

We formulate the training objective $\mathcal{L}$ as below:

$$\mathcal{L} = \mathcal{L}_1 + \lambda_{VGG}\mathcal{L}_{VGG} + \lambda_{seg}\mathcal{L}_{seg},$$

where λ_{VGG} and λ_{seg} are positive values for rebalancing each term in the training objectives. The first term $\mathcal{L}_1$ measures the difference between the rendered image $\tilde{\mathcal{I}}$ and the ground truth image I_{gt}:

$$\mathcal{L}_1 = ||\tilde{\mathcal{I}} - I_{gt}||_1.$$

To enhance the rendering fidelity and achieve better convergence on $\mathcal{L}_1$, we add a second term of perceptual loss as

$$\mathcal{L}_{VGG} = \sum_i ||VGG_i(\tilde{\mathcal{I}}) - VGG_i(I_{gt})||_1,$$

where $VGG_i(\cdot)$ indicates extracting the intermediate feature from the ith layer of a pretrained VGG network. The last term $\mathcal{L}_{seg}$ is segmentation loss,

$$\mathcal{L}_{seg} = ||\mathcal{M} - M_{gt}||_1,$$

which is the L_1 distance between the rendered mask $\mathcal{M}$ and the ground truth segmentation mask M_{gt}.

To mitigate overfitting, we perform data augmentation while training. In order to mimic the noise pattern in the input point cloud, we randomly jitter each point in the point cloud $\boldsymbol{q}$ with a gaussian noise. We find that data augmentation helps stabilize the training.

Figure 6.2: **Novel View Synthesis.** Rendering results on the holdout views of the training identities. We compare our method with KeypointNeRF [58] and Cao *et al.* [9]. Our method is compatible with different hair geometries and captures the detailed volumetric texture of varied hairstyles.

Figure 6.3: **Novel View Synthesis.** Rendering results on the test identities. We compare our method with KeypointNeRF [58] and Cao *et al.* [9]. Our method generalizes reasonably to new identities and is capable of generating a photorealistic appearance without any finetuning.

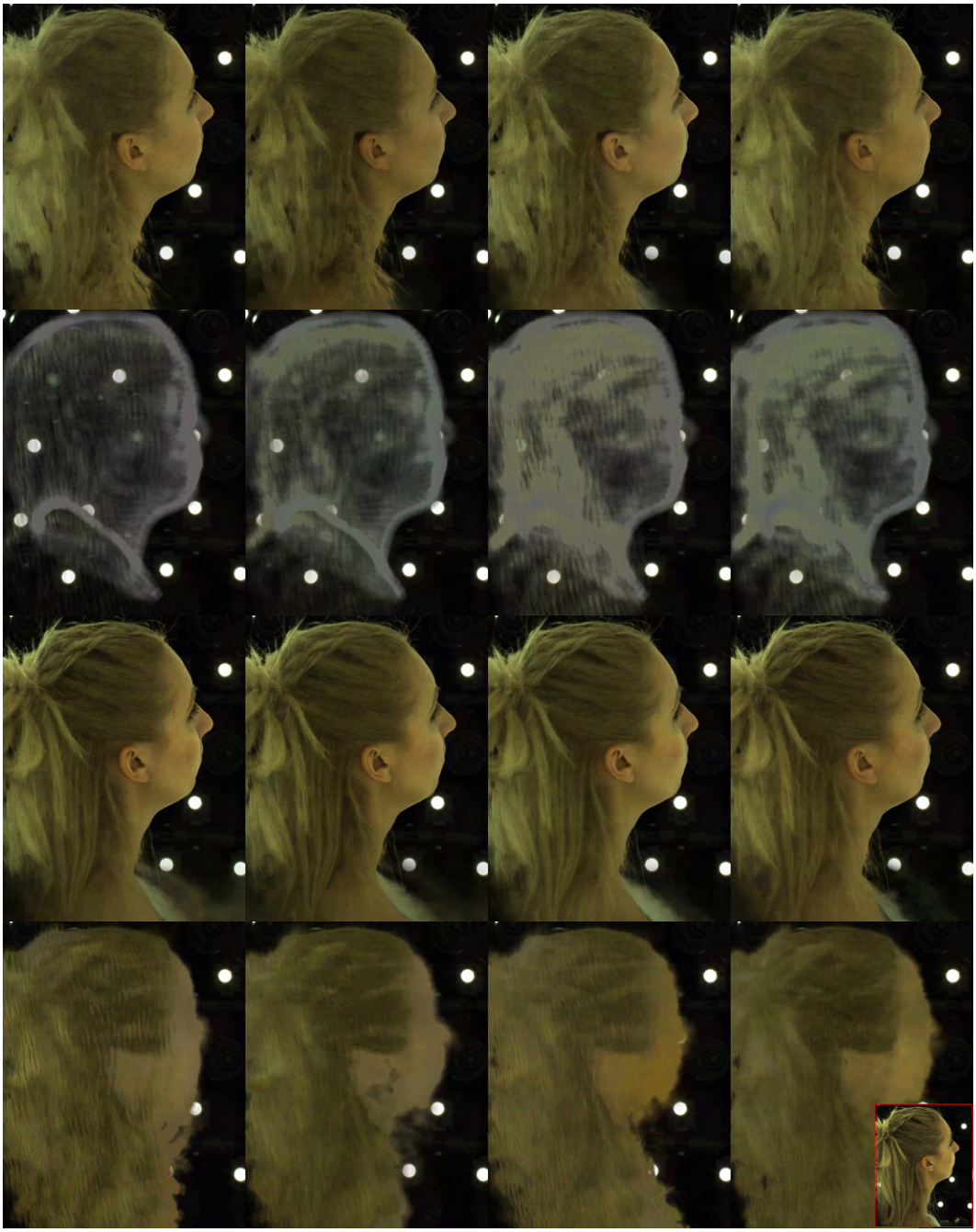

Figure 6.4: **Rendering results under different finetuning steps and views.** We show finetuned results under iteration 0 and 100 on the first and third columns respectively and train from scratch results on the second and fourth. From left to right, the results are from models trained using 10, 20, 40 and 80 views. In the lower right corner, we show the ground truth image under the rendering view for reference.

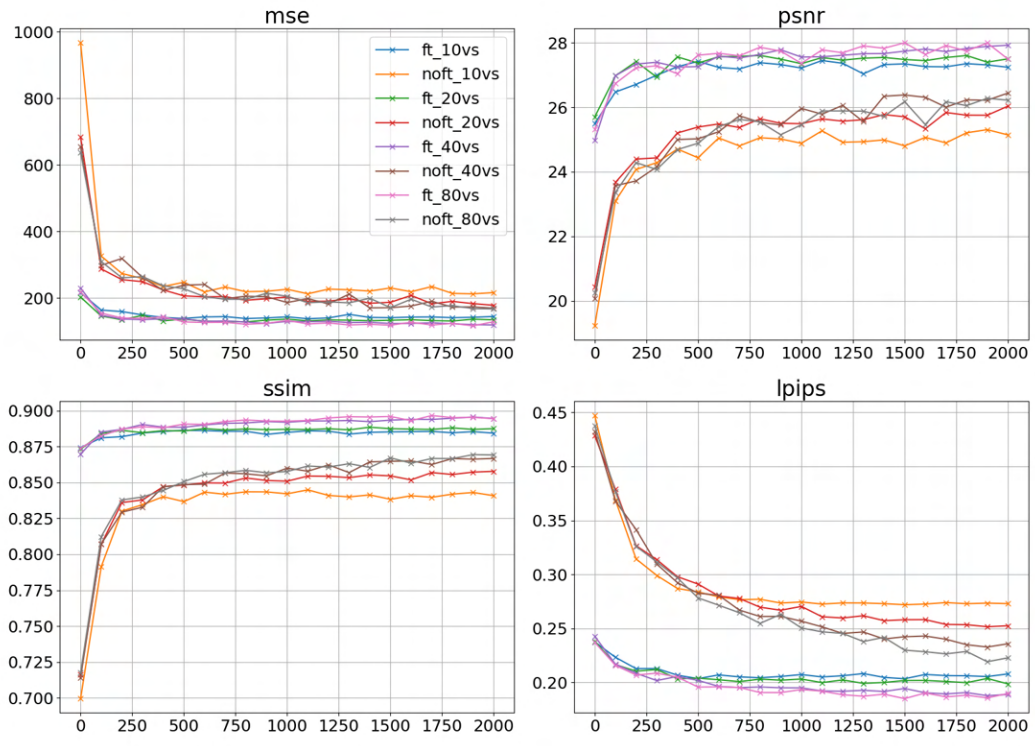

Figure 6.5: **Ablation on different finetuning configurations.** We show the learning curve of models under different finetuning configurations. We finetune(ft) our model as well as train from scratch(noft) with a varied number of training views in $\{10, 20, 40, 80\}$ that are approximately uniformly sampled from all training views. Our pre-trained model creates a warm start for avatar personalization and is also robust to the number of views used for finetuning.

Figure 6.6: **Rendering results on iPhone captured data.** We show the results of our method and instant-ngp on the iPhone-captured data. Both our method and instant-ngp work well on the training views while our method works better on testing views

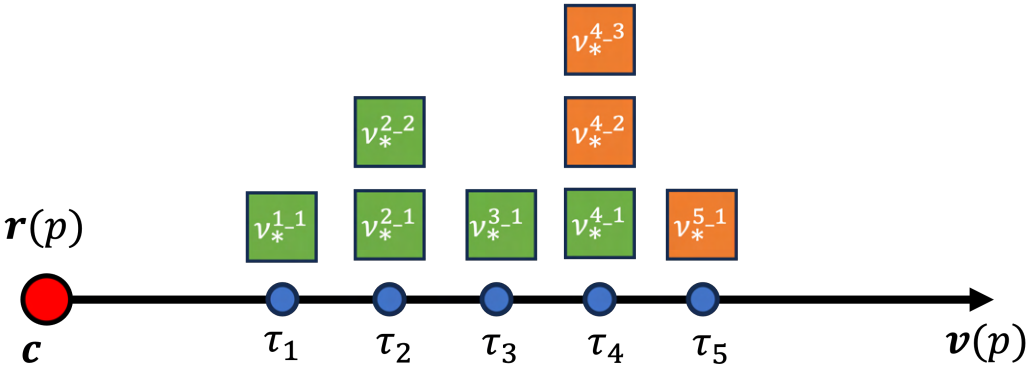

Figure 6.7: **Raymarching example.**

Chapter 7

Conclusion and Future Work

In this thesis, we have studied the problem of the dynamic capture of the human head with dynamic hair from videos. There are three major components of the captured content: geometry, appearance and motion. We first explore the use of a neural volumetric representation for a photorealistic appearance capture of near-static hair. We then further extend the problem to capture hair with motion using a hybrid neural volumetric representation that improves the efficiency of rendering as well as the accuracy of dynamic capture. Combined with a Decoder-as-a-Tracker (DaaT) algorithm, we achieve robust and automatic capture of dynamic hair with the help of both data and model priors. In addition to the dynamic capture of hair, we studied the problem of hair animation without relying on instant hair observations as a driving signal. Powered by the data from the capture stage, we build a data-driven animation model that can propagate an initial hair configuration into future configuration recurrently, while having the generated hair motion be consistent with the head motion and relative gravity direction. In parallel with building an identity-specific model for detailed dynamic capture of individual hairstyles, we tackle the problem of efficient and generalizable capture of diverse hairstyles with a single universal hair appearance model. To handle the large intra-class variance in hair appearance and geometry, we learn a local appearance prior model at the primitive level that achieves reasonable generalization on novel hairstyles with a limited amount of data. Beyond the work we have done, we believe that there is much more that can be done to improve capture efficiency and accuracy as well as generalization power for animation and content creation. In the following, we will further discuss the lessons we learned and our outlook on future directions.

1. **Dynamic Capture and Animation.** In the computer vision community, researchers attempted to solve the problem of dynamic capture under ill-posed conditions like incomplete observations, with the help of different kinds of

priors. This problem of perception aims at the accurate reconstruction of real-world assets that go beyond incomplete observations. In contrast, conventional graphics research is focused on the creation of visual data and animation, which is a generation problem. Although they are inverse to each other, the similarity between the perception and generation problems is the use of a prior model that can guide both processes. In this thesis, we jointly solve the problems of dynamic capture (perception) and animation (generation) of humans in an end-to-end manner. The benefit of solving the two problems together goes beyond the consideration of automating the pipeline. We believe that the synergy between perception and generation is key to asymptotically approaching an ideal prior model that both aligns well with observations of the world and is capable of generating realistic and diverse content. The problems of how to build and drive such a prior model are equally important for human avatars and virtual telepresence. In this thesis, we build a data-driven dynamic model of hair that propagates a previous hair state into a future one. However, the motion is limited by the training data distribution and does not cover very complex motion types. One way to solve this problem is using a more advanced sequential model such as transformer [97] and including more diverse data for training.

2. **Model-based vs. Data-driven Prior.** In this thesis, we have demonstrated the importance of both a model-based prior and a data-driven prior for the performance capture of dynamic hair. Using a model-based prior, we can explicitly regulate the model and restrict the solution space for better optimization efficiency and convergence. However, it is usually not straightforward to register an existing model prior to the input observations. One reason is that existing models might be too complex to be aligned given the incomplete form of the observations. Another reason is that some phenomena might go beyond the capability of existing models. Thus, we develop data-driven models, which are trained in both a supervised and unsupervised manner, to explain certain observations from a statistical perspective. With the model-based priors as constraints, the data-driven model is more physically grounded and converges faster during the learning process. Although this kind of model might not be fully explainable and could be limited by the training data distribution, we believe that it serves as a stepping stone towards building better model-based priors and models that are more precisely aligned with observations. One possible future direction is to use a differentiable simulator to solve for a plausible joint distribution of physical parameters via data-driven optimization. Another interesting future direction is to use a learning-based model as a guide for improving existing physical

models.

3. **Layered Representation.** Much of the work in the literature uses a unified model to jointly capture and animate the head and hair. In our work, we argue that using a layered representation instead provides many benefits beyond a unified model. The first benefit it brings us is compositionality. Although we might see the co-existence of a certain hairstyle and a certain identity from a dataset, theoretically the same hairstyle can be put on anyone's head. Compared with a unified model, a layered representation treats hair and head as separate modules, and therefore is less affected by the co-existence bias introduced by certain datasets. Additionally, a layered representation leads to better accuracy in modeling dynamics and appearance. With a layered representation, we will be able to use hair-specific prior models, rather than a prior model of scalp, to fit the specific hair dynamic and apparel properties. However, this kind of design requires that we manually determine how many layers to use and which parts should be modeled separately. Future work could automatically determine which parts need to be separately modeled and how many layers are needed. Besides the head, many hairstyles interact frequently with shoulders and hands. Good modeling for each of those body parts is essential for a natural and realistic appearance. Furthermore, as those parts enjoy a smaller intra-class variance than hair among different identities, the interaction between those parts and hair will serve as a good clue for anchoring the shape and dynamics property of different hairstyles.

4. **Hybrid Representation for Efficient Rendering and Animation.** A 3D representation that is efficient to render and store is essential for animation. Our methods used a hybrid representation for that purpose. The hybrid representation contains a coarse-level explicit geometric representation that can be controlled directly and a fine-level representation of a neural volumetric function which is stored with a sparse structure. Although that can be made quite efficient to drive and render, there is still room to improve. One promising direction is to use a mipmap-like structure to store the appearance at different levels, such that we can adaptively select the content to render according to the resolution needs. Another possible improvement is to generate the neural appearance of the visible parts rather than all parts, improving the run-time efficiency.

5. **Generalization and Scalability.** Building a generalizable and scalable model is essential for democratizing high-quality human avatars. Toward this goal, many research directions have been explored in this thesis such as finding

suitable neural 3D representation for dynamic human modeling, efficient inverse rendering techniques for distilling 3D avatars from 2D images and building a data-driven dynamic model for better control of human avatars. Besides the technical improvements, data is fuel for making those models work. We detail the scale of the data we used in each chapter in Tab 7.1 and discuss what model we build with those data. We hope it could inspire future data collection for generalizable high-quality avatars. From Chap. 3 to Chap. 5 we build identity-specific models for various motion patterns like expressions and hair movements. In Chap. 3, we build a volumetric model for capturing static hair. Although the data we used are videos, there is little hair motion variation. In Chap. 4, we capture videos with hair movements and use a sparse volumetric model for detailed capture of hair movements. However, the videos we captured contain a single motion and are not enough to build animatable avatars. Therefore, in Chap. 5, we further expand the data variation in our dataset by introducing more hair motions captured under different head velocities. The newly captured dataset facilitates a more robust dynamic model that can generate diverse hair motions. In Chap. 6, the size of our dataset drops dramatically in terms of frames but increases significantly in the number of identities covered. We explore a generalizable local appearance model for hair appearance capture for a large group of individuals with diverse hairstyles. In the future, we hope to build an avatar that generalizes on both appearance and dynamics. To achieve that, we can collect a new dataset that exhausts all combinations of different appearances and motions. However, the efforts for creating such a dataset would be tremendous. To avoid that, we need to disentangle the appearance and motion from real-world observations and treat them as tangential parts of our avatar modeling. Then, we will be able to use a collection of much smaller datasets that capture the variance only in appearance or motion.

There are two major challenges to building a more universal model for both appearance and dynamics. The first challenge is the great intra-class variance in the appearance and geometry of human hair. With a limited amount of data for individuals, this problem becomes even more challenging. Most of the existing large-scale datasets are noisy and unstructured, which makes them hard to use for building a generalizable model with high quality. How to utilize existing large-scale internet data to upgrade existing prior models is an interesting and challenging problem. One possible approach is to utilize large foundation models that are trained on internet data as a critique to optimize existing prior models for better generalization on appearance. Another difficulty is on the dynamic modeling side. Although visual correlation can

	Chap. 3	Chap. 4 (HVH)	Chap. 5 (NeuWigs)	Chap. 6
Number of frames	360	360	1800	255
Types of hair motion	0	1	6	0
Number of identities	1	1	1	255

Table 7.1: **Datasets used in each chapter.** We compare the number of frames, types of hair motion and number of identities we captured for the datasets we used in each chapter.

sometimes indicate the similarity in dynamic patterns for different objects, it might fail to do so, especially when we have no observation of the internal structure or are tricked by objects made of different materials but with similar textures. Thus a model based only on visual conditioning might not be sufficient for a generalizable dynamic model. How to incorporate inputs from other modalities and how to leverage appropriate physics-based models are interesting directions for future research. For example, we can use additional multi-modal input such as language to help determine the dynamic properties and materials of an object.

Looking forward, we believe that unifying perception and generation is a good approach to solving other dynamic human modeling problems and beyond. To accommodate the need for modeling more sophisticated real-world scenarios, we shall break the limitations of existing models and iteratively update the model using novel data-driven priors from observations. This data-driven optimization ensures the model's accuracy from a statistical perspective. A key component in this loop is the scheme of analysis-by-synthesis with differentiable pathways, which seamlessly registers the internal model and the real-world observation. To go beyond observations and approximate the true underlying distribution of the dynamic world, we need to develop a deep understanding of physically grounded mechanisms. We believe that the two directions of perception and generation will become more integrated and the synergy between them will spur more advanced development in both computer vision and graphics communities.

Bibliography

[1] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky. Neural point-based graphics. In *European conference on computer vision*, pages 696–712. Springer, 2020.

[2] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *European Conference on Computer Vision*, pages 441–459. Springer, 2020.

[3] T. Bagautdinov, C. Wu, T. Simon, F. Prada, T. Shiratori, S.-E. Wei, W. Xu, Y. Sheikh, and J. Saragih. Driving-signal aware full-body avatars. *ACM Transactions on Graphics*, 40(4):1–17, 2021.

[4] F. Bertails, S. Hadap, M.-P. Cani, M. Lin, T.-Y. Kim, S. Marschner, K. Ward, and Z. Kačić-Alesić. Realistic hair simulation: animation and rendering. In *ACM SIGGRAPH 2008 classes*, pages 1–154. 2008.

[5] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual Conference on Computer Graphics and Interactive Techniques*, pages 187–194, 1999.

[6] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021.

[7] G. Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[8] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics*, 39(4):86–1, 2020.

[9] C. Cao, T. Simon, J. K. Kim, G. Schwartz, M. Zollhoefer, S.-S. Saito, S. Lombardi, S.-E. Wei, D. Belko, S.-I. Yu, Y. Sheikh, and J. Saragih. Authentic volumetric avatars from a phone scan. *ACM Transactions on Graphics*, 41(4), jul 2022.

[10] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*. Springer, 2020.

[11] M. Chai, J. Ren, and S. Tulyakov. Neural hair rendering. In *European Conference on Computer Vision*, pages 371–388. Springer, 2020.

[12] M. Chai, C. Zheng, and K. Zhou. A reduced model for interactive hairs. *ACM Transactions on Graphics*, 33(4):1–11, 2014.

[13] M. Chai, C. Zheng, and K. Zhou. Adaptive skinning for interactive hair-solid simulation. *IEEE transactions on visualization and computer graphics*, 23(7):1725–1738, 2016.

[14] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5799–5809, 2021.

[15] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[16] W. Chen, H. Ling, J. Gao, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[17] 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. In *European Conference on Computer Vision*. Springer, 2016.

[18] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[19] G. Gafni, J. Thies, M. Zollhöfer, and M. Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8649–8658, June 2021.

[20] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[21] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[22] P.-W. Grassal, M. Prinzler, T. Leistner, C. Rother, M. Nießner, and J. Thies. Neural head avatars from monocular rgb videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18653–18664, June 2022.

[23] P. Guan, L. Sigal, V. Reznitskaya, and J. K. Hodgins. Multi-linear data-driven dynamic hair model with efficient hair-body collision handling. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*, pages 295–304, 2012.

[24] Y. Hong, B. Peng, H. Xiao, L. Liu, and J. Zhang. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20374–20384, June 2022.

[25] L. Hu, D. Bradley, H. Li, and T. Beeler. Simulation-ready hair capture. In *Computer Graphics Forum*, volume 36, pages 281–294. Wiley Online Library, 2017.

[26] L. Hu, C. Ma, L. Luo, and H. Li. Robust hair capture using simulated examples. *ACM Transactions on Graphics*, 33(4):1–10, 2014.

[27] H. Iben, M. Meyer, L. Petrovic, O. Soares, J. Anderson, and A. Witkin. Artistic simulation of curly hair. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 63–71, 2013.

[28] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[29] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[30] K. Kania, K. M. Yi, M. Kowalski, T. Trzciński, and A. Tagliasacchi. CoNeRF: Controllable Neural Radiance Fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[31] A. Kar, C. Häne, and J. Malik. Learning a multi-view stereo machine. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[32] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[35] A. Kornilova, M. Faizullin, K. Pakulev, A. Sadkov, D. Kukushkin, A. Akhmetyanov, T. Akhtyamov, H. Taherinejad, and G. Ferrer. Smartportraits: Depth powered handheld smartphone dataset of human portraits for state estimation, reconstruction and synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21318–21329, June 2022.

[36] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision*, pages 471–488. Springer, 2016.

[37] C. Lassner and M. Zollhöfer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2021.

[38] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4d scans. *ACM Transactions on Graphics*, 36(6):194–1, 2017.

[39] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022.

[40] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021.

[41] S. Lin, L. Yang, I. Saleemi, and S. Sengupta. Robust high-resolution video matting with temporal guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 238–247, 2022.

[42] D. B. Lindell, J. N. Martel, and G. Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565, 2021.

[43] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.

[44] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[45] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics*, 37(4), July 2018.

[46] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics*, 38(4), July 2019.

[47] S. Lombardi, T. Simon, G. Schwartz, M. Zollhoefer, Y. Sheikh, and J. Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics*, 40(4), July 2021.

[48] M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*. Springer, 2014.

[49] T. Luan, L. Feng, and L. Xiaoming. Towards high-fidelity nonlinear 3d face morphoable model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[50] L. Luo, H. Li, S. Paris, T. Weise, M. Pauly, and S. Rusinkiewicz. Multi-view hair capture using orientation fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1490–1497. IEEE, 2012.

[51] L. Luo, H. Li, and S. Rusinkiewicz. Structure-aware hair capture. *ACM Transactions on Graphics*, 32(4):1–12, 2013.

[52] L. Luo, C. Zhang, Z. Zhang, and S. Rusinkiewicz. Wide-baseline hair capture using strand-based refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 265–272, 2013.

[53] Q. Lyu, M. Chai, X. Chen, and K. Zhou. Real-time hair simulation with neural interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[54] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, 2013.

[55] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[56] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[57] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6878–6887, 2019.

[58] M. Mihajlovic, A. Bansal, M. Zollhoefer, S. Tang, and S. Saito. KeypointNeRF: Generalizing image-based volumetric avatars using relative spatial encoding of keypoints. In *European conference on computer vision*, 2022.

[59] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 38(4), 2019.

[60] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 38(4):1–14, 2019.

[61] 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. In *European Conference on Computer Vision*. Springer, 2020.

[62] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[63] G. Nam, C. Wu, M. H. Kim, and Y. Sheikh. Strand-accurate multi-view hair capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 155–164, 2019.

[64] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[65] K. Olszewski, D. Ceylan, J. Xing, J. Echevarria, Z. Chen, W. Chen, and H. Li. Intuitive, interactive beard and hair synthesis with generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7446–7456, 2020.

[66] S. Paris, H. M. Briceno, and F. X. Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics*, 23(3):712–719, 2004.

[67] S. Paris, W. Chang, O. I. Kozhushnyan, W. Jarosz, W. Matusik, M. Zwicker, and F. Durand. Hair photobooth: geometric and photometric acquisition of real hairstyles. *ACM Transactions on Graphics*, 27(3):30, 2008.

[68] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[69] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[70] 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. *arXiv preprint arXiv:2106.13228*, 2021.

[71] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*. Springer, 2020.

[72] L. Petrovic, M. Henne, and J. Anderson. Volumetric methods for simulation and rendering of hair. *Pixar Animation Studios*, 2(4):1–6, 2005.

[73] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[74] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[75] A. Raj, M. Zollhofer, T. Simon, J. Saragih, S. Saito, J. Hays, and S. Lombardi. Pixel-aligned volumetric avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11733–11742, June 2021.

[76] C. Reiser, S. Peng, Y. Liao, and A. Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021.

[77] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[78] D. Rückert, L. Franke, and M. Stamminger. Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (TOG)*, 41(4):1–14, 2022.

[79] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391. IEEE, 2008.

[80] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz. Persistent point feature histograms for 3d point clouds. In *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pages 119–128, 2008.

[81] S. Saito, J. Yang, Q. Ma, and M. J. Black. Scanimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2886–2897, 2021.

[82] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[83] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.

[84] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[85] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[86] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.

[87] T. Sun, G. Nam, C. Aliaga, C. Hery, and R. Ramamoorthi. Human Hair Inverse Rendering using Multi-View Photometric data. In A. Bousseau and M. McGuire, editors, *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association, 2021.

[88] R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 517–524. IEEE, 1998.

[89] J. Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp.

[90] A. Tewari, F. Bernard, P. Garrido, G. Bharaj, M. Elgharib, H.-P. Seidel, P. Pérez, M. Zöllhofer, and C. Theobalt. Fml: Face model learning from videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10812–10822, 2019.

[91] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. In *The Proceeding of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[92] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 1274–1283, 2017.

[93] J. Thies, M. Zollhöfer, and M. Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics*, 38(4), 2019.

[94] L. Tran and X. Liu. Nonlinear 3d face morphable model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7346–7355, 2018.

[95] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021.

[96] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, volume 29, 2017.

[97] 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.

[98] C. Wang, B. Eckart, S. Lucey, and O. Gallo. Neural trajectory fields for dynamic novel view synthesis. *arXiv preprint arXiv:2105.05994*, 2021.

[99] Z. Wang, T. Bagautdinov, S. Lombardi, T. Simon, J. Saragih, J. Hodgins, and M. Zollhofer. Learning compositional radiance fields of dynamic human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, June 2021.

[100] Z. Wang, T. Bagautdinov, S. Lombardi, T. Simon, J. Saragih, J. Hodgins, and M. Zollhofer. Learning compositional radiance fields of dynamic human heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5704–5713, 2021.

[101] Z. Wang, G. Nam, T. Stuyck, S. Lombardi, C. Cao, J. Saragih, M. Zollhöfer, J. Hodgins, and C. Lassner. Neuwigs: A neural dynamic model for volumetric hair capture and animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8641–8651, 2023.

[102] Z. Wang, G. Nam, T. Stuyck, S. Lombardi, M. Zollhoefer, J. Hodgins, and C. Lassner. Hvh: Learning a hybrid neural volumetric representation for dynamic hair performance capture, 2021.

[103] K. Ward, F. Bertails, T.-Y. Kim, S. R. Marschner, M.-P. Cani, and M. C. Lin. A survey on hair modeling: Styling, simulation, and rendering. *IEEE transactions on visualization and computer graphics*, 13(2):213–234, 2007.

[104] Y. Wei, E. Ofek, L. Quan, and H.-Y. Shum. Modeling hair from multiple views. In *ACM SIGGRAPH 2005 Papers*, pages 816–820. 2005.

[105] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[106] C. Wu, T. Shiratori, and Y. Sheikh. Deep incremental learning for efficient high-fidelity face tracking. *ACM Transactions on Graphics*, 37(6):1–12, 2018.

[107] C. Wu and T. Kanai. Data-driven detailed hair animation for game characters. *Computer Animation and Virtual Worlds*, 27(3-4):221–230, 2016.

[108] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[109] C.-h. Wuu, N. Zheng, S. Ardisson, R. Bali, D. Belko, E. Brockmeyer, L. Evans, T. Godisart, H. Ha, A. Hypes, et al. Multiface: A dataset for neural face rendering. *arXiv preprint arXiv:2207.11243*, 2022.

[110] W. Xian, J.-B. Huang, J. Kopf, and C. Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021.

[111] D. Xiang, F. Prada, C. Wu, and J. Hodgins. Monoclothcap: Towards temporally coherent clothing capture from monocular rgb video. In *2020 International Conference on 3D Vision (3DV)*, pages 322–332. IEEE, 2020.

[112] Z. Xu, H.-T. Wu, L. Wang, C. Zheng, X. Tong, and Y. Qi. Dynamic hair capture using spacetime optimization. *ACM Transactions on Graphics*, 33(6), nov 2014.

[113] G. Yang and D. Ramanan. Volumetric correspondence networks for optical flow. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[114] L. Yang, Z. Shi, Y. Zheng, and K. Zhou. Dynamic hair modeling from monocular videos using deep neural networks. *ACM Transactions on Graphics*, 38(6):1–12, 2019.

[115] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021.

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

[117] K. Zhang, F. Luan, Q. Wang, K. Bala, and N. Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021.

[118] Q. Zhang, J. Tong, H. Wang, Z. Pan, and R. Yang. Simulation guided hair dynamics modeling from video. In *Computer Graphics Forum*, volume 31, pages 2003–2010. Wiley Online Library, 2012.

[119] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[120] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.

[121] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13545–13555, June 2022.

[122] Y. Zheng, W. Yifan, G. Wetzstein, M. J. Black, and O. Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21057–21067, 2023.

[123] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4), 2018.