Towards Real-time Controllable Neural Face Avatars

Heng Yu

CMU-RI-TR-23-52

July 26, 2023

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

Thesis Committee:
László A. Jeni, *chair*
Srinivasa Narasimhan
Shubham Tulsiani
Mosam Dabhi

Submitted in partial fulfillment of the requirements for the degree of Masters of Science in Robotics.

To youth.

Abstract

Neural Radiance Fields (NeRF) are compelling techniques for modeling dynamic 3D scenes from 2D image collections. These volumetric representations would be well suited for synthesizing novel facial expressions but for three problems. First, deformable NeRFs are object agnostic and model holistic movement of the scene: they can replay how the motion changes over time, but they cannot alter it in an interpretable way. Second, controllable volumetric representations typically require either time-consuming manual annotations or 3D supervision to provide semantic meaning to the scene. Third, classic NeRF-based methods rely on numerical integration which involves sampling hundreds of points across the ray, and evaluating the MLP at all of those locations, making them prohibitively slow for real-time applications.

In this work, we propose a real-time controllable neural representation for face self-portraits, that solves all of these problems within a common framework, and it can rely on automated processing. We use automated facial action recognition (AFAR) to characterize facial expressions as a combination of action units (AU) and their intensities. AUs provide both the semantic locations and control labels for the system. We also extend the light field network, the re-formulations of radiance fields to oriented rays, to dynamic de-formations and hyperspace representations to accelerate the rendering speed. Our method outperforms competing methods for novel view and expression synthesis in terms of visual and anatomic fidelity of expressions, and also achieves an order of magnitude faster rendering speed than state-of-the-art methods.

Acknowledgments

I would like to express my deepest appreciation to my advisor, Dr. László A. Jeni, for his unwavering guidance, support, and mentorship throughout my Master's study journey. Dr. Jeni's exceptional expertise, insightful feedback, and dedication to my academic and personal growth have been instrumental in shaping this work. His patience, encouragement, and belief in my abilities have been a constant source of inspiration. I am truly grateful for the invaluable lessons I have learned under his tutelage. Without him, this work would not have been possible.

I am also immensely grateful to my collaborators, Joel Julin, Zoltán Á. Milacski, and Koichiro Niinuma, for their valuable contributions, constructive discussions, and tireless efforts in advancing our research. Their unique perspectives, expertise, and camaraderie have enriched my understanding, broadened the scope of my work, and made the research journey enjoyable and fulfilling.

Furthermore, I extend my sincere appreciation to the members of my thesis committee, Prof. Srinivasa Narasimhan, Prof. Shubham Tulsiani, and Mosam Dabhi, for their insightful feedback, valuable suggestions, and critical evaluation of my research. Their expertise and diverse perspectives have played a crucial role in shaping the final outcome of this thesis. I am indebted to them for their time, effort, and guidance throughout the review process.

I would also like to thank the members of our research lab, CUBE, for their invaluable support, collaboration, and stimulating discussions. Their contributions, whether technical or moral, have been indispensable to the completion of this work. I am grateful for the collaborative environment fostered within our lab, which has provided a nurturing platform for personal and intellectual growth.

Lastly, I want to acknowledge the support of my friends and family, who have been my pillars of strength and a constant source of encouragement throughout this challenging endeavor. Their unwavering belief in my abilities, understanding of the sacrifices I had to make, and their unwavering support have been crucial in keeping me motivated and focused.

Funding

This research was supported partially by Fujitsu.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

3D understanding of the world plays a vital role in driving the next wave of technological innovations, particularly in the realm of creating digital representations of scenes, objects, and humans. While 2D images provide valuable visual information, they lack the depth and spatial awareness that are inherent in 3D data. However, acquiring 3D supervision for systems that rely on 3D understanding can be prohibitively expensive compared to their 2D counterparts.

Traditionally, capturing 3D data requires specialized equipment such as depth sensors, LiDAR, or structured light systems, which can be costly and time-consuming to set up. Additionally, annotating and labeling 3D data for training machine learning models is a labor-intensive task that further increases the cost. As a result, leveraging 3D supervision for training large-scale systems becomes a challenging and expensive endeavor.

To address these limitations, researchers have been exploring alternative approaches that can infer 3D information solely from 2D image collections. One notable technique is the use of neural volumetric representations, with Neural Radiance Fields (NeRF) [42] being a compelling example. NeRF is a neural network-based model that learns to represent 3D scenes by capturing both geometry and appearance information from a set of 2D images. It can generate high fidelity 3D reconstructions that are visually consistent with the observed 2D images.

The key idea behind NeRF is to model a continuous 3D scene as a function that maps 3D spatial coordinates to radiance values. By training a neural network on

a collection of 2D images, NeRF learns to approximate this underlying function, effectively enabling the synthesis of novel views from arbitrary viewpoints within the scene. This approach bypasses the need for explicit 3D supervision, as the model directly learns the 3D scene representation from 2D image data.

The advantages of using neural volumetric representations like NeRF for building high fidelity 3D models from 2D image collections are numerous. Firstly, it significantly reduces the cost and complexity associated with acquiring 3D supervision. Since NeRF operates solely on 2D images, it eliminates the need for specialized 3D capture equipment, saving both time and resources. Moreover, annotating large-scale 3D datasets becomes more manageable, as it only requires collecting a diverse set of 2D images.

Additionally, neural volumetric representations provide greater flexibility and generality in handling complex scenes, objects, and humans. They can capture intricate geometric details and appearance variations that are challenging to represent accurately with traditional 3D models. By leveraging the expressive power of neural networks, these representations can produce highly realistic and visually consistent 3D reconstructions.

Neural face avatars are a specific application domain where the use of neural volumetric representations has gained significant attention. Creating realistic digital representations of human faces is a challenging task due to the complex geometry and appearance variations involved. Traditional 3D modeling approaches often require laborious manual sculpting, intricate mesh rigging, and texture mapping. However, neural volumetric representations offer an alternative avenue for generating high-fidelity face avatars.

In this work, our goal is to build a real-time controllable neural face avatar system that can achieve high-fidelity 3D reconstruction and control of complex facial movements using a simplified camera setup and minimal manual annotation.

In this chapter, we provide an introduction to our work, starting with the motivation behind our research in Section 1.1. Subsequently, in Section 1.2, we delve into the challenges that we aim to address within this research area. In Section 1.3, we present a concise summary of the contributions made by our research. To provide a roadmap for the rest of the work, we outline the structure and content in Section 1.4. Through these sections, we aim to establish the context, significance,

and contributions of our research, while also providing a clear roadmap for readers to follow as they explore the subsequent chapters of this work.

1.1 Motivation

The field of computer vision, computer graphics, and virtual reality (VR) has witnessed tremendous advancements in recent years, enabling highly realistic and immersive virtual environments. However, the creation of lifelike digital avatars that accurately represent human facial expressions and movements remains a significant challenge. Traditional approaches to avatar creation often require labor-intensive manual rigging, complex animation pipelines, and costly motion capture systems. These limitations hinder the real-time control and responsiveness necessary for interactive applications such as virtual communication, gaming, and film production.

Therefore, our motivation is to develop a real-time controllable neural face avatar system that overcomes the limitations of traditional methods. By leveraging the power of the neural rendering method, we aim to create a novel framework capable of generating high-fidelity facial avatars that accurately mimic human expressions.

The primary objective of this work is to enable intuitive and natural control over the facial avatar, empowering users to manipulate its appearance and behavior in real time. By providing users with direct and responsive control, we aim to enhance the sense of presence and immersion in virtual environments, improving the overall user experience.

Furthermore, the proposed system seeks to reduce the dependency on costly hardware and complex capture setups. By utilizing a simplified camera setup (just a cellphone) and fully automatic annotation, we aim to democratize the creation of facial avatars, making them more accessible to a wider range of users and applications.

The potential applications of a real-time controllable neural face avatar system are vast and diverse. From virtual communication platforms that enable users to represent themselves with lifelike avatars during remote interactions to interactive storytelling experiences where digital characters can be controlled and animated in real-time, the impact of this research can revolutionize the way we interact with virtual worlds.

Ultimately, the development of a real-time controllable neural face avatar system

has the potential to reshape the fields of VR, computer vision, computer graphics, and human-computer interaction. By bridging the gap between digital avatars and real-life facial expressions, this research opens up new possibilities for entertainment, communication, education, and various other domains.

1.2 Challenges

Building a real-time controllable neural face avatar system poses several significant challenges that need to be addressed to achieve the desired outcomes. These challenges are crucial to consider in order to develop a robust and effective system:

Data Availability and Annotation Obtaining high-quality training data for neural face avatar systems is a challenge in itself. Collecting diverse and representative facial expression datasets that cover a wide range of motions, lighting conditions, and demographics can be time-consuming and resource-intensive. Additionally, manual annotation of the data to provide ground truth information for training can be challenging due to the subjective nature of facial expressions and the need for meticulous annotation efforts.

Anatomically Correct Control The achievement of anatomically correct control in high-fidelity 3D modeling of facial appearance and dynamics poses a significant challenge in the field. Previous approaches based on neural representations for facial actions have either relied on parametric models to encode facial expressions or exhibited limitations in the level of control over scene attributes.

Real-time Performance Another challenge is achieving real-time performance for the neural face avatar system. Generating high-fidelity facial animations in real time requires efficient algorithms and optimized computational techniques. The system must process input data, perform complex computations, and render the results within strict time constraints to maintain interactive frame rates. Balancing accuracy and responsiveness is a critical challenge to ensure a smooth and immersive user experience.

1.3 Contributions

Addressing key challenges in the development of a real-time controllable neural face avatar system, the contributions of this work are as follows:

Fully Automatic Annotation We achieve fully automatic annotation of facial data by applying automated facial action recognition (AFAR) to characterize facial expressions as a combination of action units (AU) and their intensities. AUs provide both the semantic locations and control labels for the system. This contribution eliminates the need for labor-intensive manual annotation, enabling more efficient data processing and alleviating the burden of annotation tasks.

High Fidelity 3D Reconstruction and Control We present an innovative neural representation that enables both high-fidelity 3D reconstruction and precise control of intricate facial movements, all achieved with a simplified camera setup. Our approach achieves a fully disentangled representation in the feature space, where distinct semantic regions are independent of each other. Furthermore, each region is equipped with multiple semantic control variables, allowing for fine-grained control over specific aspects of the facial expression.

Real-Time Performance Additionally, another significant contribution of this work is the implementation of real-time rendering capabilities throughout the entire system by applying the idea of Light Field Networks (LFNs)(LFNs) [59], allowing for the generation of facial animations in real time. This achievement ensures that the system can deliver seamless and immediate visualization of the rendered facial expressions, enhancing the interactive and immersive nature of the user experience.

1.4 Outline

This report is organized as follows:

- **Chapter 2: Controllable Neural Face Avatars** In this chapter, we explore the methodology behind achieving controllability of neural face avatars through the use of automatic annotation. We delve into the details of how automatic annotation techniques can be employed to enable precise control over the facial expressions and attributes of the avatars.

- **Chapter 3: Making Light Field Networks Dynamic** In this chapter, our focus will be on discussing the techniques and methodologies employed to achieve real-time neural rendering. We will delve into the details and explore the innovative approaches used to enable the generation and visualization of neural-rendered outputs in real time.
- **Chapter 4: Real-time Controllable Neural Face Avatars** In this chapter, our objective is to elucidate the process of building real-time controllable neural face avatars by leveraging the techniques introduced in the preceding chapters. We will delve into the implementation details and explore how these techniques are integrated to construct a system that achieves both real-time performance and precise control over facial avatars.

This work is based on the following articles:

- Yu, et al. "CoNFies: Controllable Neural Face Avatars." 2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG). IEEE, 2023.
- Yu, et al. "DyLiN: Making Light Field Networks Dynamic." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

Chapter 2

Controllable Neural Face Avatars

In this chapter, we will delve into the techniques and methodologies employed to achieve the controllability of neural face avatars. Neural Radiance Fields (NeRF) [42] are compelling techniques for modeling dynamic 3D scenes from 2D image collections. These volumetric representations would be well suited for synthesizing novel facial expressions but for two problems. First, deformable NeRFs are object agnostic and model holistic movement of the scene: they can replay how the motion changes over time, but they cannot alter it in an interpretable way. Second, controllable volumetric representations typically require either time-consuming manual annotations or 3D supervision to provide semantic meaning to the scene. We propose a controllable neural representation for face self-portraits (CoNFies), that solves both of these problems within a common framework, and it can rely on automated processing. We use automated facial action recognition (AFAR) to characterize facial expressions as a combination of action units (AU) and their intensities. AUs provide both the semantic locations and control labels for the system. CoNFies outperformed competing methods for novel view and expression synthesis in terms of visual and anatomic fidelity of expressions.

2.1 Introduction

3D understanding of the world is crucial to the next round of technological innovations in creating digital representations of scenes, objects, and humans. However, the cost

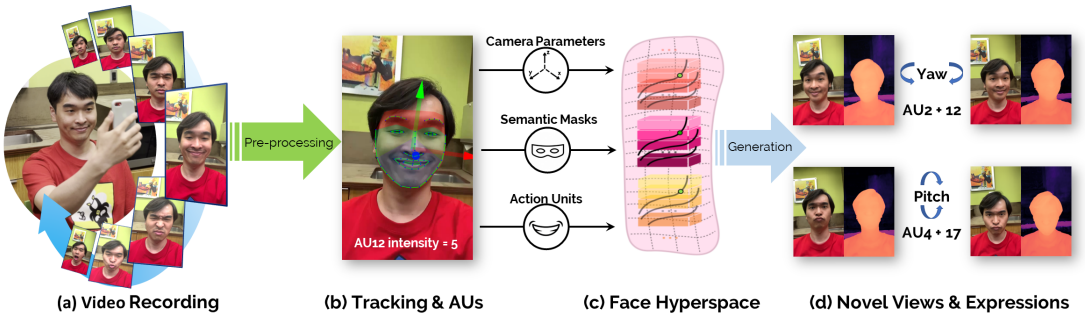


Figure 2.1: 2D video of users recording themselves using a circular motion during a semi-structured facial expressions task (a) is processed by a person-independent face tracker that codes facial action units (AU) (b). The estimated camera parameters, semantic masks, action unit intensities, and the original 2D frames are used to build a disentangled face hyper-space (c). From this representation, novel views and unseen expressions can be generated along with their 3D depth (d).

of getting 3D supervision for such systems is astronomically higher than those in 2D. Neural volumetric representations, such as Neural Radiance Fields (NeRF) [42], are compelling alternatives for building high-fidelity representations from 2D image collections only.

Although, previous work in this direction has demonstrated promising results for modeling and synthesizing novel views of static scenes [75] [20] [13] [72], articulated objects [63] [52] [21], have explored the use of coarse-grain controls over limited properties, such as color [25], material [77], and object editing [73], relatively neglected is the fine-level control of semantic scene attributes.

Our interest is in general social interactions, and thus we are interested in high-fidelity 3D modeling of facial appearance and dynamics. Previous neural representation based approaches for facial actions either required parametric models to encode facial expressions [21] [4] or were limited in the level of control over scene attributes [51].

In a recent work Kania et al. [27] proposed a controllable neural representation that can achieve simple manipulation, such as opening and closing the mouth and the eyes using a learned mapping between a segmentation mask and a control value that describes the state of that region. The method relies on temporally sparse and manual annotation of the regions of interest along with their control signals, which limits the application of the method.

We wish to make high-fidelity 3D reconstruction and control of complex facial movements with a simplified camera setup and little or no manual annotation. A high-level summary of our method is shown on Fig. 2.1. First, we capture fine-scale transitions of facial movements during a semi-structured expression task. The recorded video is then processed with an automated facial action recognition system [18] [7] that provides anatomically correct action unit (AU) [56] intensities and facial landmarks. From these, semantic facial masks are generated automatically and frames are sub-sampled to build an AU-balanced set of training data. The selected 2D frames, semantic masks, AU intensities, and camera parameters then used to build a face hyper-space, that can be used to synthesize novel views and unseen expression combinations.

Our contributions are as follows:

- **Anatomically Correct Control.** Previous work was limited to holistic deformations or required manual annotation. We achieve an anatomically controllable neural avatar with no manual annotation. We show that this is achievable by using automated facial action coding that provides consistent facial key-points and semantic labels across subjects.
- **Multi-label Semantic Masks.** We achieve a completely disentangled representation in the feature space where the different semantic regions do not affect each other and each region has multiple semantic control variables. We demonstrate that this formulation correctly handles different action unit combinations and achieves better visual fidelity than previous methods.

2.2 Related Works

Our work focuses on automatic control over avatar expressions and is closely related to several computer vision and graphics research domains such as neural rendering and avatar animation.

2.2.1 Neural Rendering and Novel View Synthesis

Implicit neural representations represented by NeRF has become more and more popular recently. NeRF can synthesize high-quality rendering results from novel views

and some following extensions further enhanced the algorithm in rendering quality improvement [75] [72], faster training and inference [22] [74] [45], generalization model [58] [47] and so on. NeRF and its variants achieve remarkable performance on static objects, and several of these variants extend it to dynamic scenes, which is the same scenario as ours. Park et al. [50] [51] introduce deformation fields along with a canonical NeRF to learning the movements. Some other works handle the dynamic scene through learning movement offset [53] [63] or scene flow [34] [71]. These methods achieve eye-catching results in dynamic scenes and can achieve the separation of static parts and dynamic parts to some extent through the learned deformation field/offset/flow. However, they are far from the fine-grained control over the dynamic scenes.

2.2.2 Avatar Animation

Avatar animation is a well explored research area. Some works have attempted to manipulate or edit a face [17] [31] [61] while they are mainly image-based and fail to leverage 3d representation. Other works [30] [28] [62] utilize 3D Morphable Model (3DMM) as 3D face representation to achieve the head pose control and image or video reanimation. However, they have limited ability to synthesize novel views since they neglect scene geometry or appearance. Given that high-quality novel view synthesis and fine-grained avatar control is pretty challenging, some works take advantage of neural rendering to model a non-rigid 3d avatar. NerFACE [21] allows face expression and head pose control by modeling a 4D face avatar using neural radiance fields and a facial expression tracking algorithm while it assumes a static background and fixed camera. Some other works either require professional equipment and training dataset [39] or impose parametric face models such as 3DMM [3] [4], which limit their application scenarios. HyperNeRF [51] introduces hyper space that can better fit dynamic face avatars and also control facial expressions to some extent through hyper space. However, it is far from fine-grained control and cannot achieve per-attribute control. CoNeRF [27] can achieve per-attribute control by imposing an attribute value and mask based on HyperNeRF while it can only achieve simple control over each attribute and different attributes may affect each other. It also requires manual labeling which is labor-intensive. In contrast, we propose an

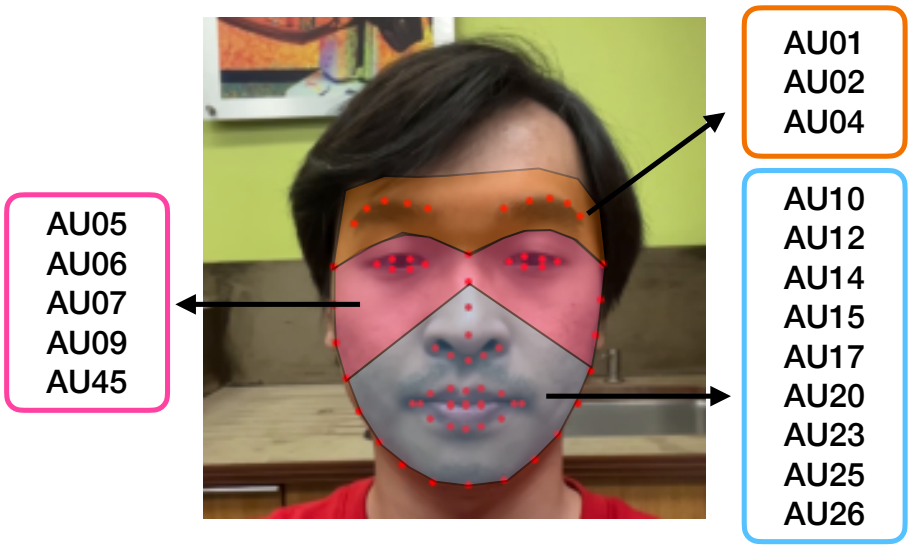


Figure 2.2: Facial landmarks, AUs and mask annotations

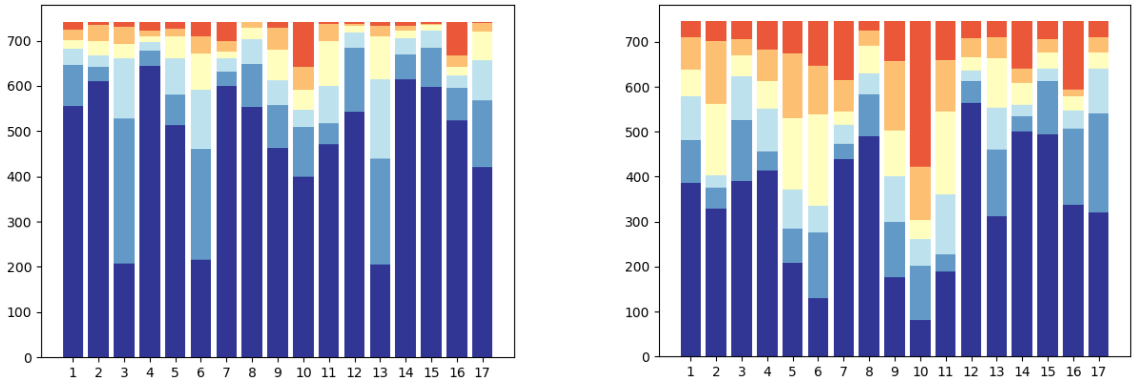
automatic system that enables fine-grained comprehensive control over a face avatar and novel-view synthesis simultaneously without any manual labels. Recently, Cao et al. [12] proposed an approach to creating volumetric avatars using only a short phone capture. Though their approach can generate a high-fidelity avatar, a large-scale high-resolution multi-view dataset is required to pre-train their model. Unlike their approach, our method requires only a single input video.

2.3 Method

Our system consists of three parts: (i) data and annotation processing, (ii) network training, and (iii) avatar control. We will describe each component in detail in the following parts.

2.3.1 Data and annotation processing

The data collection process of our system can be done using just a smart phone with a slo-mo video function. After collecting slo-mo video of changing expressions with moving cameras, we apply OpenFace [7] on every frame to detect the facial landmarks [2] [5] and facial action units (AUs) [6]. It is worth noting that other facial



(a) Uniform Sampling (b) Balanced Sampling

Figure 2.3: AU intensity distribution using different sampling strategies

landmark/AU detection methods can also work and may have better results, but this is not the focus of our work. OpenFace can output 68 2D landmark locations and 17 AU intensities from 0 to 5 (as shown in Fig. 2.2). To mitigate the noise impact of AUs detection between adjacent frames, we apply Savitzky–Golay filter [57] to smooth the AUs and then sample the frames to reduce the computational load of the whole system. We found uniform sampling can lead to an extreme imbalance in AU intensity distribution (Fig. 2.3(a)) since there exist many neutral frames in the dataset. To alleviate this problem, we propose a balanced sampling strategy. We define each AU value and AU intensity pair as a AU-intensity block, which consists of frames with a corresponding AU value and intensity. We ascendingly sort all the AU intensity blocks (total block number equals to intensity number times AU number) according to frame number in each block first. Then we uniformly sample frames from each block in order. Before sampling each block, we remove the frames that are already sampled. Using this strategy, we can get a more balanced sampling result shown in Fig. 2.3b.

After obtaining the 2D facial landmarks and AUs, we generate attribute masks and controllable AUs values as annotations. We define three regions and assign each action unit (AU) to its corresponding mask as shown in Fig. 2.2. We calculate the middle points of eyebrow and eye key-points as the boundary between the first and second region. We also extend some distance along the eyebrow direction as the boundary of the first region to make sure the whole eyebrows are included in the

region. The boundary of the third region consists of landmarks (#3 − #13, #28) detected by OpenFace. We also normalize each AU according to:

$$AU' = \min(\frac{AU - AU_{min}}{\alpha AU_{max} - AU_{min}} \times 2 - 1, 1) \tag{2.1}$$

where $AU \in [0, 5]$, $AU' \in [-1, 1]$, AU_{min} and AU_{max} are minimum and maximum and values for each AU among all frames, respectively, and α is the factor that adjusts the maximum of AUs and we set α as 0.8 for all the experiments.

2.3.2 Network architecture

In this section, we briefly introduce NeRF [42], HyperNeRF [51] and CoNeRF [27] for completeness and then describe our approach in detail.

Neural Radiance Field (NeRF).

NeRF uses a fully-connected neural network to learn the implicit 3D scene volumetric representations through a partial set of 2D images and can generate novel views. The NeRF network takes a sample 3d position $\mathbf{x} = (x, y, z)$ and a 2d view direction $\mathbf{d} = (\theta, \phi)$ as input and outputs the emitted color $\mathbf{c}$ and volume density σ at position $\mathbf{x}$ with view direction $\mathbf{d}$. Then one can accumulate the densities and colors into image pixels $\mathbf{C}$ in RGB using classical volume rendering techniques [26] as follows:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), d)dt \tag{2.2}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$, $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is the camera ray with near bound t_n and far bound t_f. $C(\mathbf{r})$ is the expected image pixel color of the ray $\mathbf{r}(t)$.

HyperNeRF and CoNeRF.

Given that original NeRF can only model static scenes, HyperNeRF, which extend Nerfies [50], is proposed to model dynamic objects, especially face avatars, by introducing canonical hyper-space. The input of HyperNeRF includes sample point $\mathbf{x}$ and view direction $\mathbf{d}$, which is similar to the template NeRF, and also a latent deformation

code ω_i and a latent appearance code ψ_i. The sample point $\mathbf{x}$ concatenated with the image's latent deformation code ω_i are taken as input to the spatial deformation field T and the ambient slicing surface H as follows, which yields a warped coordinate $\mathbf{x'}$ and a coordinate in ambient space $\mathbf{w}$, respectively.

$$\mathbf{x'} = T(\mathbf{x}, \omega_i); \quad \mathbf{w} = H(\mathbf{x}, \omega_i) \tag{2.3}$$

The density σ and color $\mathbf{c}$ can be then obtained by taking $\mathbf{x'}$, $\mathbf{w}$ along with direction $\mathbf{d}$ and latent appearance code ψ_i as input into template NeRF F:

$$(\sigma, \mathbf{c}) = F(\mathbf{x'}, \mathbf{w}, \mathbf{d}, \psi_i) \tag{2.4}$$

HyperNeRF can model time-varying shapes even with topological changes and can render different expressions of face avatar by using setting specific ambient coordinates in hyper-space. However, it is far from fine-grained control and fails to achieve per attribute control. Inspired by HyperNeRF, CoNeRF introduces regressors A and M to regress the attribute α and the corresponding mask $\mathbf{m}$. The attribute α is generated from latent deformation code ω_i and then is taken as input into ambient slicing surface H to generate a coordinate in ambient space $\mathbf{w}$:

$$\alpha = A(\omega_i); \quad \mathbf{w} = H(\mathbf{x}, \alpha) \tag{2.5}$$

The corresponding mask map $\mathbf{m}$ is generated using the warped coordinate $\mathbf{x'}$ and the ambient space $\mathbf{w}$ and then is used to mask out $\mathbf{w}$:

$$\mathbf{m} = M(\mathbf{x'}, \mathbf{w}); \quad \mathbf{w'} = \mathbf{w} \odot \mathbf{m} \tag{2.6}$$

The following density and color generation is the same as HyperNeRF. CoNeRF maps attribute α to its corresponding area through mask $\mathbf{m}$ so as to achieve the control over the corresponding area when rendering from novel views by assigning attribute α with different values. However, it requires manual labeling of attribute α and the corresponding mask $\mathbf{m}$ which is labor-intensive and can only perform simple control over each area since only one attribute is related to each mask. CoNeRF can also lead to movement in another area when controlling one area since its network architecture

does not achieve complete decoupling between attributes.

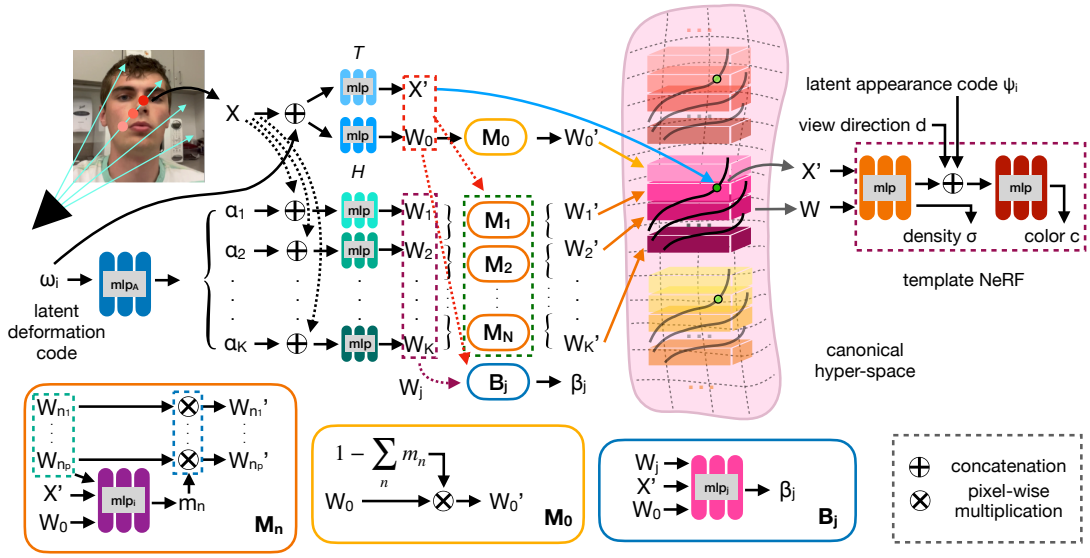


Figure 2.4: CoNFies architecture. α_i represents AU attribute learned from the latent code. β_i is uncertainty and m_i is mask.

CoNFies.

We propose CoNFies based on CoNeRF that can achieve more complex and accurate control over attributes. The network architecture of CoNFies is shown as Fig. 2.4. We learn K attributes $\alpha_{1\cdots K}$ from latent deformation code ω_i using attribute network A. Different from CoNeRF, we use *tanh* as activation function in the last layer of A to learn attributes whose range is $[-1, 1]$. We adopt the hyper-space W as proposed in HyperNeRF while our hyper-space W consists of $K + 1$ (K attributes and 1 remaining part) components, which are attribute-specific. First, K original hyper-space components are generated using X and corresponding attribute α as following:

$$\mathbf{w}_i = H_i(\mathbf{x}, \alpha_i); \quad i = 1 \cdots K \tag{2.7}$$

We also generate one hyper-space $\mathbf{w}_0$ for remaining avatar part and generate a warped coordinate $\mathbf{x'}$ through deformation field T:

$$\mathbf{w}_0 = H_0(\mathbf{x}, \omega_i); \quad \mathbf{x'} = T(\mathbf{x}, \omega_i) \tag{2.8}$$

After obtaining the original hyper-space $\mathbf{w}_{0\cdots K}$, we then learn N masks according to a pre-defined correspondence between attributes and masks (many-to-one):

$$\mathbf{m}_n = M_n(\mathbf{x'}, \mathbf{w}_0, \mathbf{w}_{n_1} \cdots \mathbf{w}_{n_p}); \quad n = 1 \cdots N \tag{2.9}$$

where $\mathbf{w}_{n_1} \cdots \mathbf{w}_{n_p}$ are the hyper-space generated from corresponding attributes $\alpha_{n_1} \cdots \alpha_{n_p}$ that are related to $\mathbf{m}_n$. The mask $\mathbf{m}_0$ for hyper-space $\mathbf{w}_0$ is $1 - \sum_n \mathbf{m}_n$. Final hyper-space $\mathbf{w}'_{0\cdots K}$ are obtained by masking the original hyper-space using corresponding masks:

$$\mathbf{w}'_i = \mathbf{w}_i \odot \mathbf{m}_j; \quad i = 1 \cdots K; \quad j = 1 \cdots N \tag{2.10}$$

where $\mathbf{m}_j$ is the mask which α_i is related to. The whole hyper-space W is obtained by concatenating $\mathbf{w}'_{0\cdots K}$ and the final density σ and color $\mathbf{c}$ are obtained using (2.4), which is the same as template NeRF. We also render the mask field into image space using an analogous volume rendering process:

$$M(\mathbf{r}|\theta, \beta_c) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{m}(\mathbf{r}(t), d)dt \tag{2.11}$$

It is also worth noting that we learn an uncertainty β_i for each attribute α_i as follows to help reduce the potential noises after AUs filtering during training.

$$\beta_i = B_i(\mathbf{x'}, \mathbf{w}_0, \mathbf{w}_i); \quad i = 1 \cdots K \tag{2.12}$$

Training Losses.

Given a training set collection of C images, the losses of our method consist of two parts, reconstruction losses L_{rec} and control losses L_{ctrl}, which are similar to [27]:

$$\arg\min_{\theta,\{\mu_c\}} L_{rec}(\theta,\{\mu_c\}) + L_{ctrl}(\theta,\{\mu_c\}) \tag{2.13}$$

where θ is network parameters and μ_c represents the latent code (deformation/appearance) of image c. Reconstruction losses L_{rec} have two parts (L_{recon} and L_{reg}). One is the primary reconstruction loss, which aims to reconstruct input observations $\{C_c\}$ as follow (gt = ground truth):

$$L_{recon} = \sum_{\mathbf{r}\in R} \|C(\mathbf{r}|\theta,\beta_c) - C^{gt}(\mathbf{r})\|_2^2 \tag{2.14}$$

The other one a Gaussian prior on the latent codes $\{\mu_c\}$ as proposed in [49]:

$$L_{reg} = \sum_{c} \|\mu_c\|_2^2 \tag{2.15}$$

Control losses L_{ctrl} also have two parts: attribute mask loss L_{mask} and attribute value loss L_{attr}, as proposed in [27]. For attribute mask loss, we first project 3D volumetric neural mask field $\mathbf{m}$ into 2D mask image using (2.11) and the attribute mask loss can be written as:

$$L_{mask} = \sum_{\mathbf{r},a} \delta_{c,a} CE(M(\mathbf{r}|\theta,\beta_c), M^{gt}_{c,a}(\mathbf{r})) \tag{2.16}$$

where $CE(\cdot,\cdot)$ represents cross entropy and $M^{gt}_{c,a}(\mathbf{r})$ is a-th attribute in the c-th image. $\delta_{c,a}$ denotes an indicator, where $\delta_{c,a} = 1$ means attribute a for image c, which $\mathbf{r}$ is belong to, is provided, otherwise $\delta_{c,a} = 0$. We also stop gradients in (2.16) w.r.t σ and employ focal loss [35] in place of the standard cross entropy loss as in [27]. For attribute value loss, we employ the AUs after filtering as ground-truth and $\beta_{c,a}$

learned from (2.12) to further reduce noises in AUs:

$$L_{attr} = \sum_{c} \sum_{a} \delta_{c,a} \frac{|\alpha_{c,a} - \alpha_{c,a}^{gt}|^2}{2\beta_{c,a}^2} + \frac{(\log \beta_{c,a})^2}{2} \tag{2.17}$$

where larger $\beta_{c,a}$ values attenuate the importance of learned $\alpha_{c,a}$ and the second term precludes the trivial minimum at $\beta_{c,a} = \infty$. Hence the network can better learn to adjust $\alpha_{c,a}$ and reduce the negative effect of noises in AUs. The final loss is:

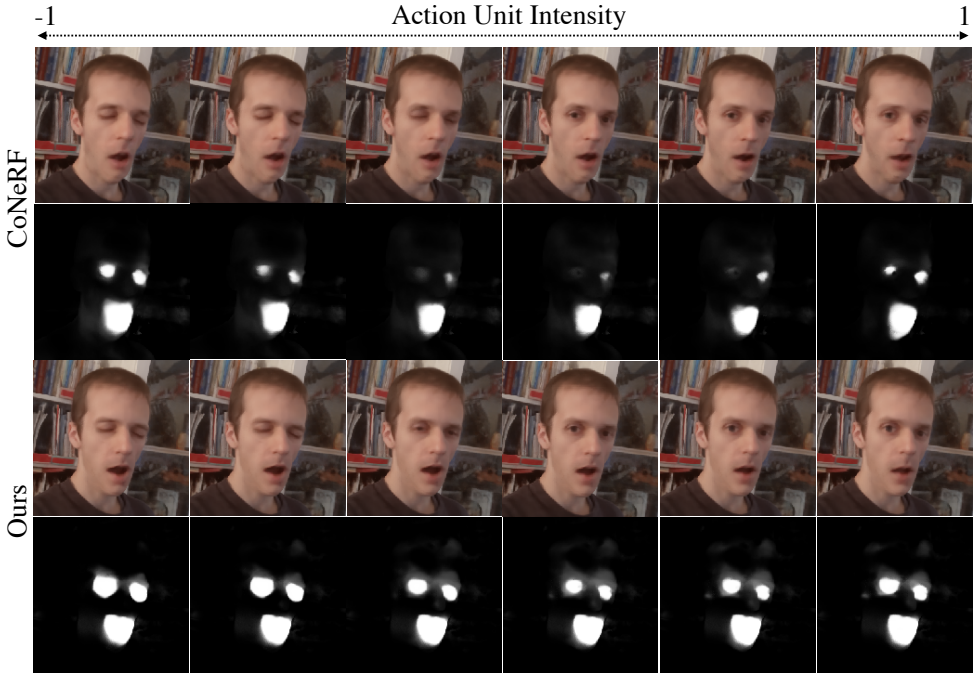
$$L = L_{recon} + w_{reg} L_{reg} + w_{mask} L_{mask} + w_{attr} L_{attr} \tag{2.18}$$

where w_{reg}, w_{mask} and w_{attr} are weighting coefficients.

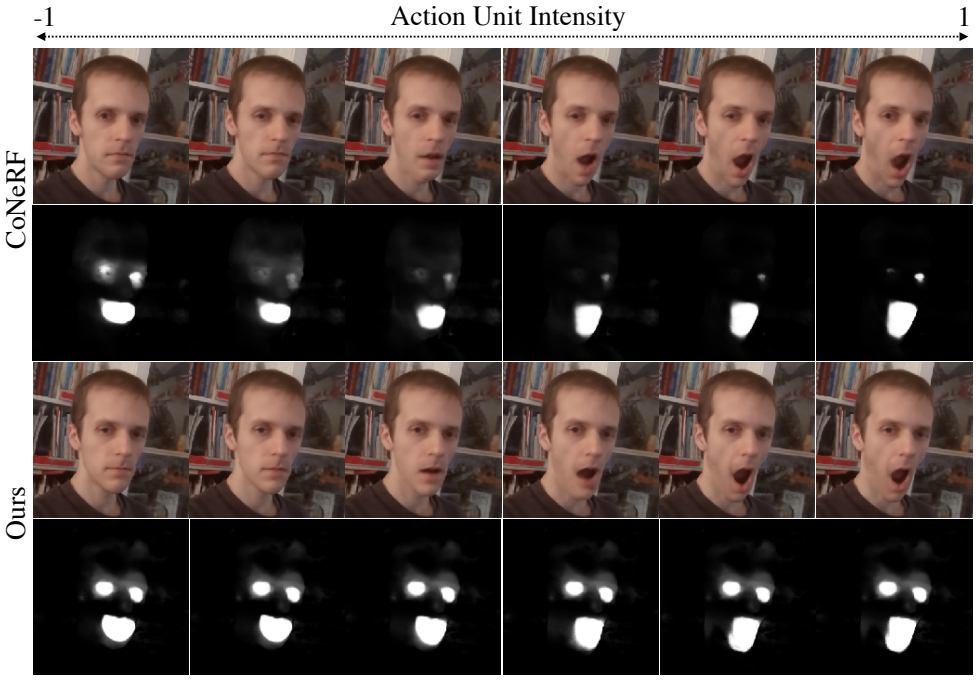
2.4 Experiments

2.4.1 Implementation details

Our method is based on the JAX [10] implementation of CoNeRF [27]. Attribute network A has six layers, each of which is a 32 neuron multi-layer perceptron (MLP) and has a skip connection at the fifth layer following [50] [51] [27]. Deformation field T and ambient slicing surface H_i have the same architecture of those in [51] [27]. Mask network M_i and uncertainty network B_i have the same structure, which is a four-layer MLP with 128 neuron per layer and followed by an additional 64 neuron layer with a skip connection as in [27]. The template NeRF is the same as original NeRF [42] but with a different input dimension size. In our case, the number of attributes K is 17, which is the number of AUs and the number of mask N is 3 as shown in Fig. 2.2. We resize all the input images to 480×270 and train our NeRF model for 250k iterations with 128 samples per ray and a batch size of 512 rays. We use Adam [29] with initial learning rate $lr = 1e-4$ and set $w_{reg} = 1e-4$, $w_{mask} = 1e-2$ and $w_{attr} = 0.1$ for all experiments. We introduce exponentially decaying on lr and w_{attr}, which decay to $1e-5$ and 0, respectively. Exponentially decaying on w_{attr} can help further reduce the noises introduced by AU detection. We train our model on a NVIDIA A100 GPU with 80G memory and the whole process takes around 26 hours



(a) eye control



(b) mouth control

Figure 2.5: Controlling results of CoNFies and CoNeRF. Our CoNFies can perform better control over one attribute without affecting other attributes.

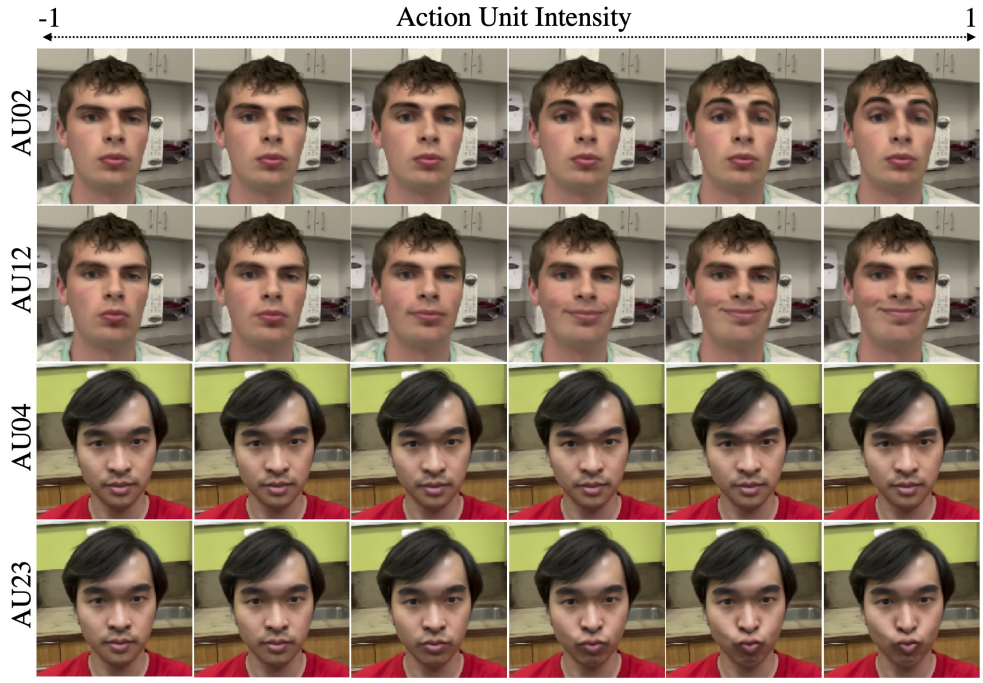


Figure 2.6: Control using single AU. AU02 is an outer brow raiser. AU04 is brow lowerer. AU12 is a lip corner puller. AU23 is a lip tightener.

2.4.2 Dataset

We used both the real dataset provided in [27] and we collected our own video sequences using a smartphone. In our data collection each of the sequences was captured with an Apple iPhone 13 Pro using 120 fps slo-mo mode and is about 2 minutes long. In each video sequence, the person performs different facial expressions related to a single AU one by one first and then performs arbitrary facial expressions related to multiple AUs. Each video is extracted to frames with 120 fps and we perform OpenFace [7] and smoothing as mentioned above. Note that OpenFace can only provide 17 AU intensities and we use these in the following experiments. More AU intensities may be obtained from other methods, which is not the focus of our paper. We then undersample the sequences to give approximately 750 frames per capture. The frames along with the AUs and attribute masks generated automatically form the datasets we use in our experiments.

Table 2.1: Intraclass Correlation (ICC) comparison between CoNeRF and our method.

AU	CoNeRF	Ours
01	0.52	0.86
02	0.54	0.73
04	0.23	0.91
05	-0.11	0.40
06	0.00	0.00
07	-0.44	0.73
09	0.55	0.74
10	0.00	0.00
12	0.00	0.87

AU	CoNeRF	Ours
14	0.17	0.77
15	-0.15	0.81
17	0.31	0.88
20	0.03	0.63
23	0.49	0.82
25	0.36	0.90
26	0.08	0.93
45	0.21	0.83
mean	0.16	0.69

2.4.3 Decoupling Mask

We compare our method with CoNeRF using their dataset with manual attribute values and mask area labels to show the effectiveness of our decoupling mask structure. We control the eyes and mouth separately using attribute values and show the rendering image results along with the masks in Fig. 2.5. We can see from the mask results that when controlling one attribute, the masks of the other attribute are not affected in our method. But one attribute can affect the others in CoNeRF, which lead to unexpected movement and artifacts in the rendering results.

2.4.4 Attribute Control

Our CoNFies can achieve attribute control using different AUs. We first show the controlling result using single AU (AU 02, 04, 12, 23) on two sequences as in Fig. 2.6. We also conducted quantitative evaluation to compare our method and CoNeRF.

In this experiment, AU intensities were obtained from synthesis images generated by our method and CoNeRF, and compare them using Intraclass Correlation (ICC). OpenFace was used to obtain AU intensities from synthesis images. Only images with extreme AU values (near -1 or 1) were manually selected to train CoNeRF while our method automatically obtained images to train the model. Table 2.1 and Fig. 2.7 show that our method outperforms CoNeRF. Fig. 2.8 compares AU01 intensity transition between control values and synthesized images. The AU intensity transition of our method is much closer to the control than CoNeRF. This result also indicates that

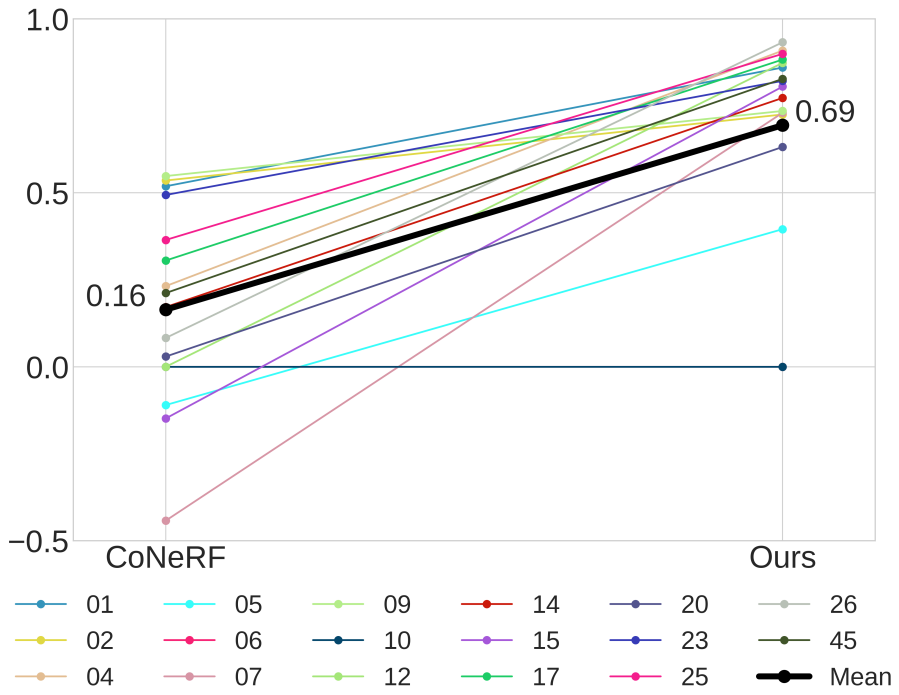


Figure 2.7: Intraclass Correlation (ICC) comparison between CoNeRF and our method.

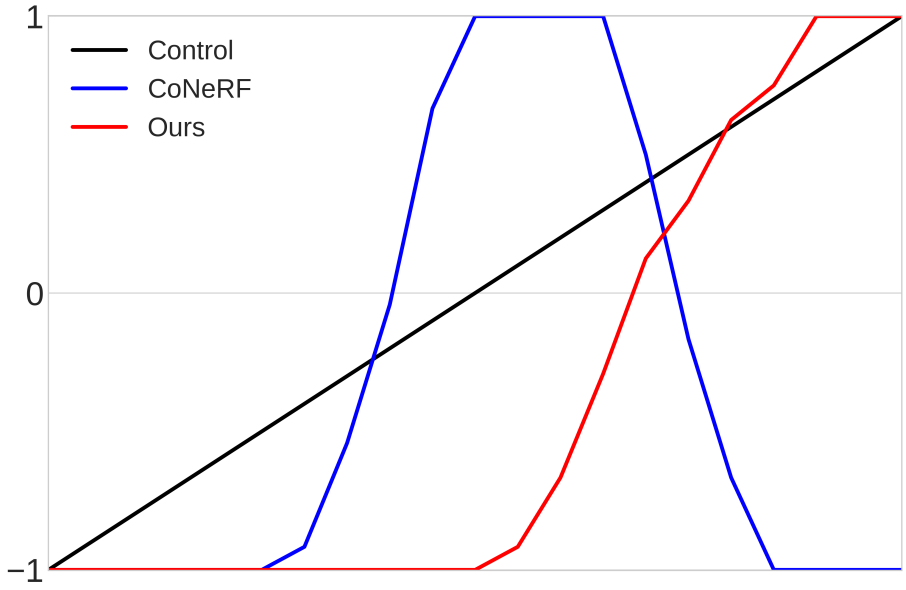


Figure 2.8: Comparison of AU01 intensity transition between control values and synthesized images.

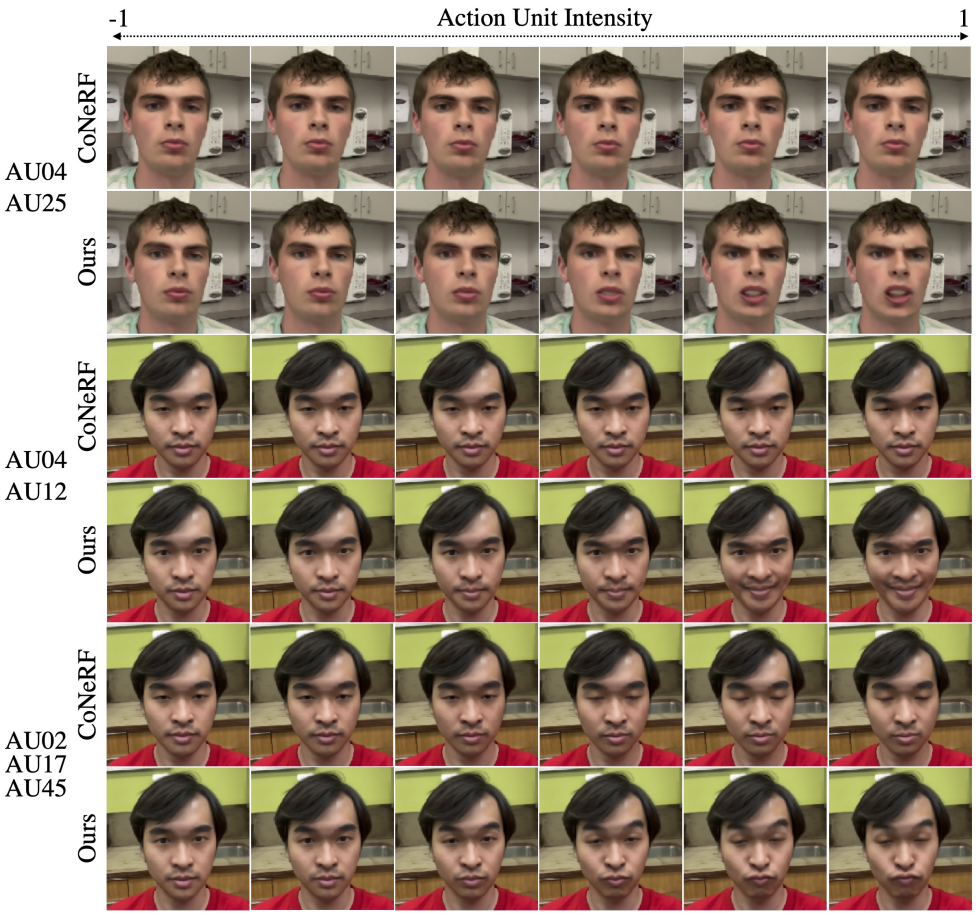


Figure 2.9: Control using multiple AUs on different regions. AU02 is outer brow raiser. AU04 is brow lowerer. AU12 is lip corner puller. AU17 is chin raiser. AU25 is lips part. AU45 is blink.

our method can handle subtle AU changes better than CoNeRF.

Our method can perform complex control using multiple AUs simultaneously as shown in Fig. 2.9 and Fig. 2.10. From Fig. 2.9, we can see that our method can perform combined AUs control over different regions, e.g. eyebrow and mouth.

We show that our method can even perform more complicated control over the same region. As shown in Fig. 2.10, we can control the smile while keeping mouth open or control mouth open while keeping the smile. However, CoNeRF cannot render good results under such complex scenarios. More visualization results can be found in the supplementary material.

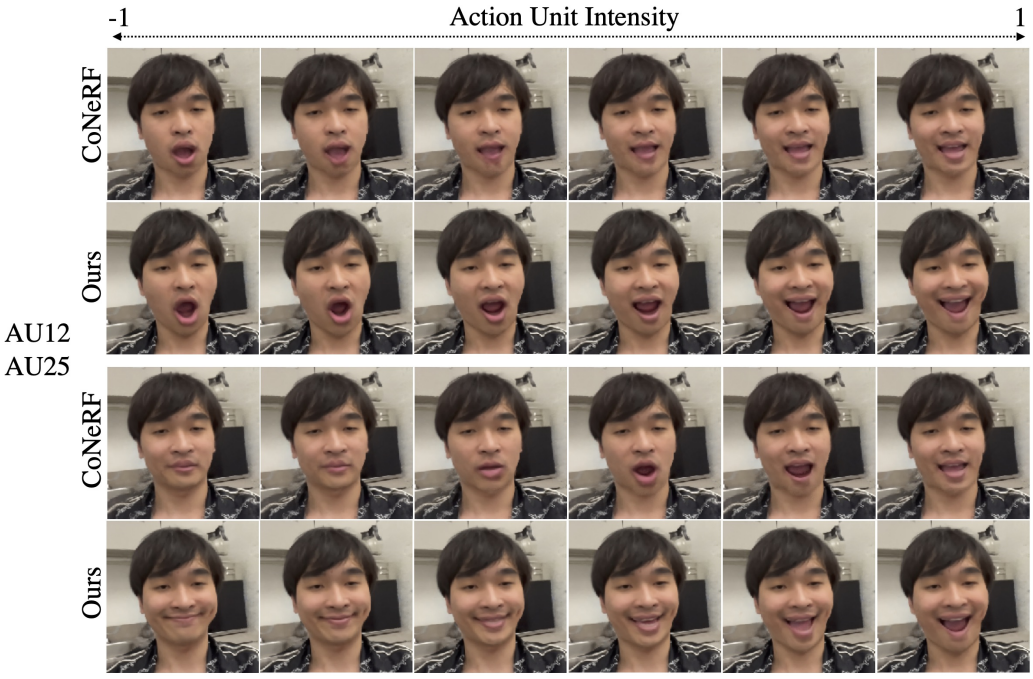


Figure 2.10: Control using multiple AUs on the same region. AU12 is lip corner puller. AU25 is lips part.

2.4.5 Novel View Synthesis

As a NeRF-based method, we can synthesize novel views with single or multiple AUs control. We show the novel view synthesis results along with corresponding masks in Fig. 2.11, where we keep the facial expression constant. To further show the power of our method, we show the rendering result under different views and control the AU values (AU02 and AU12) simultaneously in Fig. 2.12.

We also evaluate the rendering quality of our method on a frame interpolation task as proposed in [27] using the dataset it released. We interpolate every other frame and do not perform any attribute control. We use Peak Signal-to-Noise Ratio (PSNR), Multi-scale Structural Similarity (MS-SSIM) [69] and Learned Perceptual Image Patch Similarity (LPIPS) [76] to quantitatively evaluate our method compared with NeRF [42], NeRF + Latent, Nerfies [50], HyperNeRF [51], CoNeRF-M and CoNeRF [27], which is consistent with [27]. The result in shown in Table 2.2, where we can see that our method can achieve comparable performance in the rendering quality from a novel view.

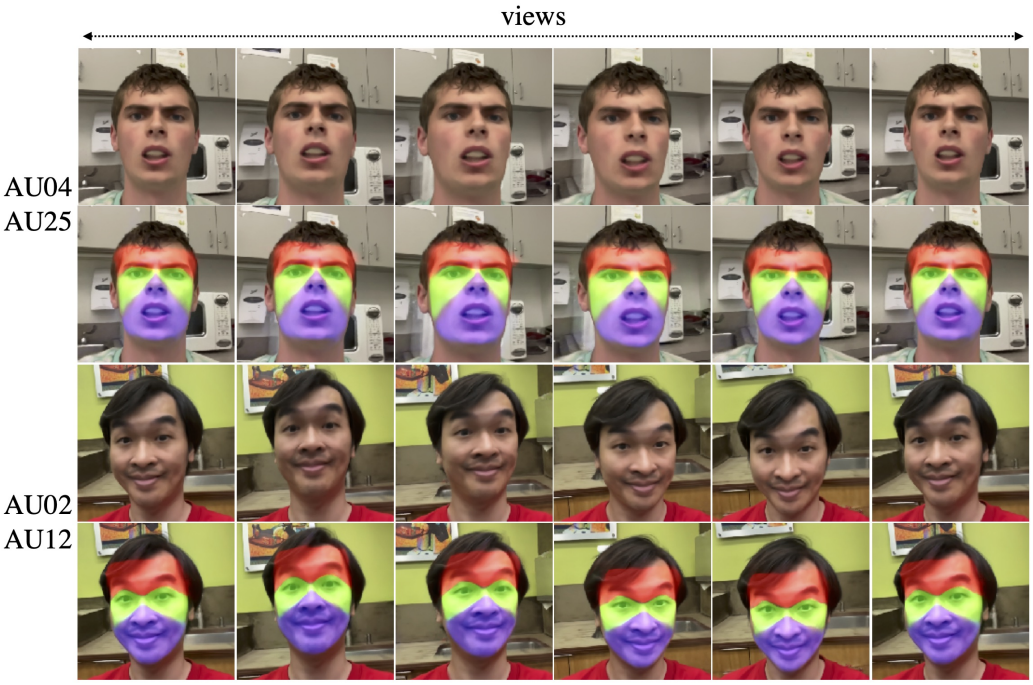


Figure 2.11: Novel view synthesis under fixed AU setting. AU02 is outer brow raiser. AU04 is brow lowerer. AU12 is lip corner puller. AU25 is lips part.

Table 2.2: Quantitative results

Method	PSNR↑	MS-SSIM↑	LPIPS↓
NeRF	28.795	0.951	0.210
NeRF + Latent	32.653	0.981	0.182
NeRFies	32.274	0.981	0.180
HyperNeRF	32.520	0.981	0.169
CoNeRF-M	32.061	0.979	0.167
CoNeRF	32.342	0.981	0.168
Ours	32.356	0.982	0.166

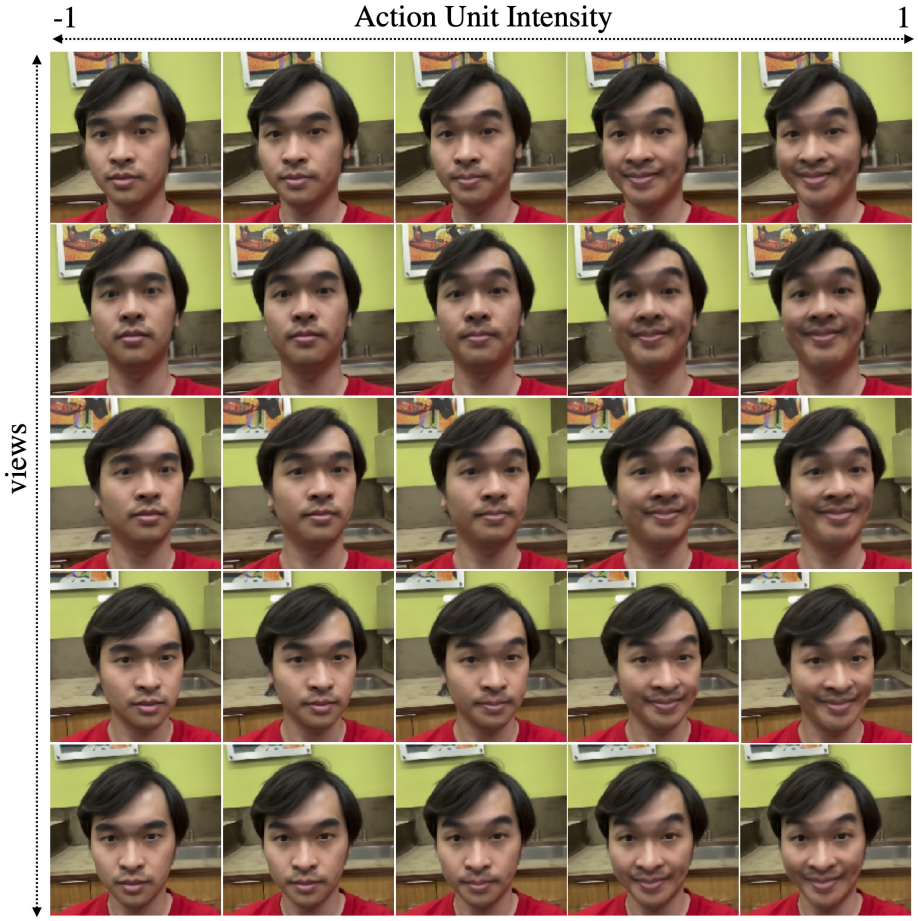


Figure 2.12: Rendering results under different AU (AU02 and AU12) intensities and views

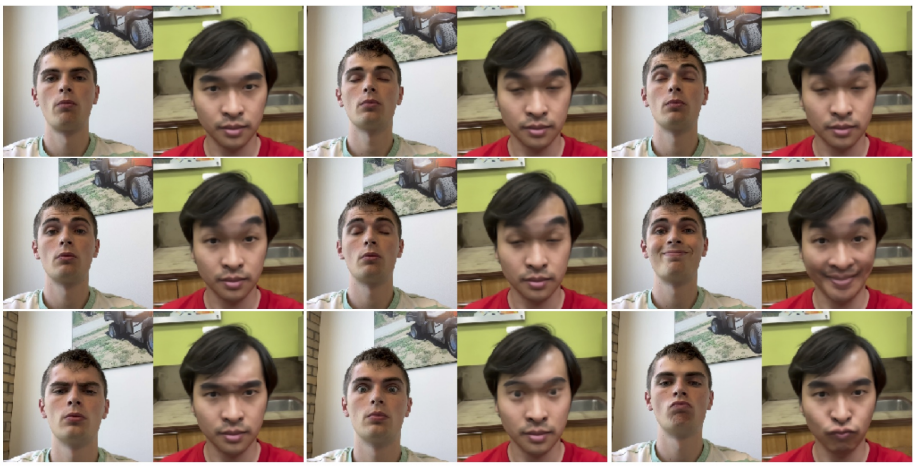


Figure 2.13: Facial expression transfer using reference sequence.

2.4.6 Facial Expression Transfer

In addition to adjusting AU values manually to control the avatar's expression, it is possible to copy them from another person's face. In this case, first we detect the AU intensities from the source person's face and use them to re-synthesize the same expression on the avatar. We show the rendering results using another face sequence to control the trained sequence in Fig. 2.13.

2.5 Conclusions

We have proposed an automated approach for controllable neural face avatars. Once a 2D video is recorded using slo-mo mode, our network is automatically trained by utilizing AU intensities and facial landmarks. We have introduced the decoupling mask structure so that the different semantic regions do not affect each other and each region have multiple control variables. We have shown that our approach outperforms CoNeRF in terms of fine-grained AU control. Moreover, our approach has a capability to control multiple AUs and novel views simultaneously.

Chapter 3

Making Light Field Networks Dynamic

In this chapter, our focus will be on discussing the methods and strategies involved in achieving real-time neural rendering. We will delve into the technical aspects and explore the innovative approaches employed to enable the generation and visualization of neural-rendered outputs in real time. Light Field Networks, the re-formulations of radiance fields to oriented rays, are magnitudes faster than their coordinate network counterparts, and provide higher fidelity with respect to representing 3D structures from 2D observations. They would be well suited for generic scene representation and manipulation, but suffer from one problem: they are limited to holistic and static scenes. In this chapter, we propose the Dynamic Light Field Network (DyLiN) method that can handle non-rigid deformations, including topological changes. We learn a deformation field from input rays to canonical rays, and lift them into a higher dimensional space to handle discontinuities. We train both models via knowledge distillation from pretrained dynamic radiance fields. We evaluated DyLiN using both synthetic and real-world datasets that include various non-rigid deformations. DyLiN qualitatively outperformed and quantitatively matched state-of-the-art methods in terms of visual fidelity, while being $25 - 71\times$ computationally faster.

3.1 Introduction

Machine vision has made tremendous progress with respect to reasoning about 3D structure using 2D observations. Much of this progress can be attributed to the emergence of coordinate networks [15, 41, 49], such as Neural Radiance Fields (NeRF) [43] and its variants [8, 40, 44, 68]. They provide an object agnostic representation for 3D scenes and can be used for high-fidelity synthesis for unseen views. While NeRFs mainly focus on static scenes, a series of works[21, 50, 53, 63] extend the idea to dynamic cases via additional components that map the observed deformations to a canonical space, supporting moving and shape-evolving objects. It was further shown that by lifting this canonical space to higher dimensions the method can handle changes in scene topology as well [51].

However, the applicability of NeRF models is considerably limited by their computational complexities. From each pixel, one typically casts a ray from that pixel, and numerically integrates the radiance and color densities computed by a Multi-Layer Perceptron (MLP) across the ray, approximating the pixel color. Specifically, the numerical integration involves sampling hundreds of points across the ray, and evaluating the MLP at all of those locations. Several works have been proposed for speeding up static NeRFs. These include employing a compact 3D representation structure [20, 37, 74], breaking up the MLP into multiple smaller networks [54, 55], leveraging depth information [16, 46], and using fewer sampling points [36, 46, 72]. Yet, these methods still rely on integration and suffer from sampling many points, making them prohibitively slow for real-time applications. Recently, Light Field Networks (LFNs) [59] proposed replacing integration with a direct ray-to-color regressor, trained using the same sparse set of images, requiring only a single forward pass. R2L [65] extended LFNs to use a very deep residual architecture, trained by distillation from a NeRF teacher model to avoid overfitting. In contrast to static NeRF acceleration, speeding up dynamic NeRFs is a much less discussed problem in the literature. This is potentially due to the much increased difficulty of the task, as one also has to deal with the high variability of motion. In this direction, [19, 66] greatly reduce the training time by using well-designed data structures, but their solutions still rely on integration. LFNs are clearly better suited for acceleration, yet, to the best of our knowledge, no works have attempted extending LFNs to the dynamic scenario.

In this paper, we propose 2 schemes extending LFNs to dynamic scene deformations, topological changes and controllability. First, we introduce DyLiN, by incorporating a deformation field and a hyperspace representation to deal with non-rigid transformations, while distilling knowledge from a pretrained dynamic NeRF. Afterwards, we also propose CoDyLiN, via adding controllable input attributes, trained with synthetic training data generated by a pretrained Controllable NeRF (CoNeRF) [27] teacher model. To test the efficiencies of our proposed schemes, we perform empirical experiments on both synthetic and real datasets. We show that our DyLiN achieves better image quality and an order of magnitude faster rendering speed than its original dynamic NeRF teacher model and the state-of-the-art TiNeuVox [19] method. Similarly, we also show that CoDyLiN outperforms its CoNeRF teacher. We further execute ablation studies to verify the individual effectiveness of different components of our model. Our methods can be also understood as accelerated versions of their respective teacher models, and we are not aware of any prior works that attempt speeding up CoNeRF.
Our contributions can be summarized as follows:

- We propose DyLiN, an extension of LFNs that can handle dynamic scenes with topological changes. DyLiN achieves this through non-bending ray deformations, hyperspace lifting for whole rays, and knowledge distillation from dynamic NeRFs.
- We show that DyLiN achieves state-of-the-art results on both synthetic and real-world scenes, while being an order of magnitude faster than the competition. We also include an ablation study to analyze the contributions of our model components.
- We introduce CoDyLiN, further extending our DyLiN to handle controllable input attributes.

3.2 Related Works

Dynamic NeRFs. NeRFs have demonstrated impressive performances in novel view synthesis for static scenes. Extending these results to dynamic (deformable) domains has sparked considerable research interest [21, 50, 51, 53, 63]. Among these

works, the ones that most closely resemble ours are D-NeRF [53] and HyperNeRF [51]. D-NeRF uses a translational deformation field with temporal positional encoding. HyperNeRF introduces a hyperspace representation, allowing topological variations to be effectively captured. Our work expands upon these works, as we propose DyLiN, a similar method for LFNs. We use the above dynamic NeRFs as pretrained teacher models for DyLiN, achieving better fidelity with orders of magnitude shorter rendering times.

Accelerated NeRFs. The high computational complexity of NeRFs has motivated several follow-up works on speeding up the numerical integration process. The following first set of works are restricted to static scenarios. NSVF [37] represents the scene with a set of voxel-bounded MLPs organized in a sparse voxel octree, allowing voxels without relevant content to be skipped. KiloNeRF [55] divides the scene into a grid and trains a tiny MLP network for each cell within the grid, saving on pointwise evaluations. AutoInt [36] reduces the number of point samples for each ray using learned partial integrals. In contrast to the above procedures, speeding up dynamic NeRFs is much less discussed in the literature, as there are only 2 papers published on this subject. Wang *et al.* [66] proposed a method based on Fourier plenoctrees for real-time dynamic rendering, however, the technique requires an expensive rigid scene capturing setup. TiNeuVox [19] reduces training time by augmenting the MLP with time-aware voxel features and a tiny deformation network, while using a multi-distance interpolation method to model temporal variations. Interestingly, all of the aforementioned methods suffer from sampling hundreds of points during numerical integration, and none of them support changes in topology, whereas our proposed DyLiN excels from both perspectives.

Light Field Networks (LFNs). As opposed to the aforementioned techniques that accelerate numerical integration within NeRFs, some works have attempted completely replacing numerical integration with direct per-ray color MLP regressors called Light Field Networks (LFNs). Since these approaches accept rays as inputs, they rely heavily on the ray representation. Several such representations exist in the literature. Plenoptic functions [1, 9] encode 3D rays with 5D representations, i.e., a 3D point on a ray and 2 axis-angle ray directions. Light fields [23, 32] use 4D ray

codes most commonly through two-plane parameterization: given 2 parallel planes, rays are encoded by the 2D coordinates of the 2 ray-plane intersection points. Sadly, these representations are either discontinuous or cannot represent the full set of rays. Recently, Sitzmann *et al.* [59] advocate for the usage of the 6D Plücker coordinate representation, i.e., a 3D point on a ray coupled with its cross product with a 3D direction. They argue that this representation covers the whole set of rays and is continuous. Consequently, they feed it as input to an LFN, and additionally apply Meta-Learning across scenes to learn a multi-view consistency prior. However, they have not considered alternative ray representations, MLP architectures or training procedures, and only tested their method on toy datasets. R2L [65] employs an even more effective ray encoding by concatenating few points sampled from it, and proposes a very deep (88 layers) residual MLP network for LFNs. They resolve the proneness to overfitting by training the MLP with an abundance of synthetic images generated by a pretrained NeRF having a shallow MLP. Interestingly, they find that the student LFN model produces significantly better rendering quality than its teacher NeRF model, while being about 30 times faster. Our work extends LFNs to dynamic deformations, topological changes and controllability, achieving similar gains over the pretrained dynamic NeRF teacher models.

Knowledge Distillation. The process of training a student model with synthetic data generated by a teacher model is called Knowledge Distillation (KD) [11], and it has been widely used in the vision and language domains [14, 32, 64, 67] as a form of data augmentation. Like R2L [65], we also use KD for training, however, our teacher and student models are both dynamic and more complex than their R2L counterparts.

3.3 Methods

In this section, we present our solution for extending LFNs. We propose DyLiN, supporting dynamic deformations and hyperspace representations via two respective MLPs. We use KD to train DyLiN with synthetic data generated by a pretrained dynamic NeRF teacher model.

3.3.1 Network Architecture

Our overall DyLiN architecture G_ϕ is summarized in fig. 3.1. It processes rays instead of the widely adopted 3D point inputs as follows.

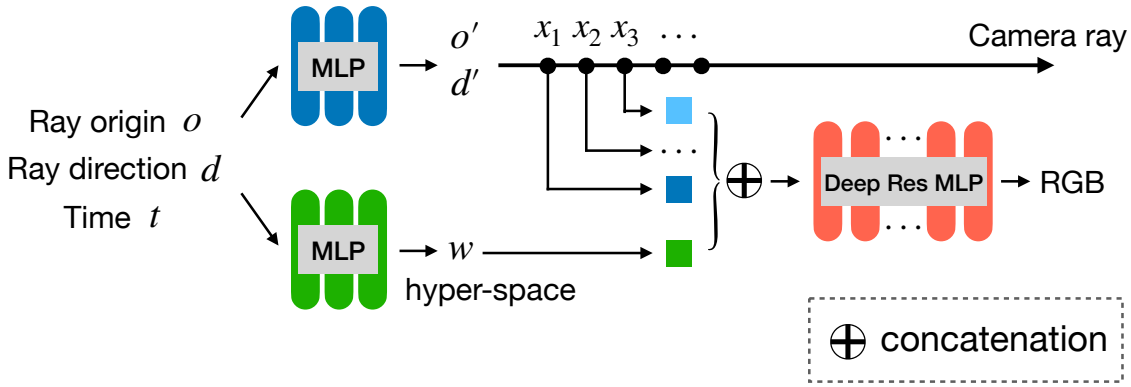


Figure 3.1: Schematic diagram of our proposed DyLiN architecture. We take a ray $r = (o, d)$ and time t as input. We deform r into $r' = (o', d')$, and sample few points x_k, $k = 1, \ldots, K$ along r' to encode it (blue). In parallel, we also lift r and t to the hyperspace code w (green), and concatenate it with each x_k. We use the concatenation to regress the RGB color of r at t directly (red).

Specifically, our deformation MLP T_ω maps an input ray $r = (o, d)$ to canonical space ray $r' = (o', d')$:

$$(o', d') = T_\omega(o, d, t). \tag{3.1}$$

Unlike the pointwise deformation MLP proposed in Nerfies [50], which bends rays by offsetting their points independently, our MLP outputs rays explicitly, hence no ray bending occurs. Furthermore, after obtaining r', we encode it by sampling and concatenating K points along it.

Our hyperspace MLP H_ψ is similar to T_ω, except it outputs a hyperspace representation w:

$$w = H_\psi(o, d, t). \tag{3.2}$$

In contrast to HyperNeRF [51], which predicts a hyperspace code w for each 3D point, we use rays and compute a single w for each ray.

Both MLPs further take the index t as input to encode temporal deformations.

Once the K points and w are obtained, we concatenate them and feed the result

into our LFN R_π, which is a deep residual color MLP regressor. Overall, we can collect the model parameters as $\phi = [\omega, \psi, \pi]$.

Note that without our two MLPs T_ω and H_ψ, our DyLiN falls back to the vanilla LFN.

3.3.2 Training Procedure

Our training procedure is composed of 3 phases.

First, we pretrain a dynamic NeRF model F_θ (e.g., D-NeRF [53] or HyperNeRF [51]) by randomly sampling time t and input ray r, and minimizing the Mean Squared Error (MSE) against the corresponding RGB color of monocular target video I:

$$\min_{\theta} \mathbb{E}_{t,r=(o,d)} \left[\| F_\theta(o,d,t) - I(o,d,t) \|_2^2 \right]. \tag{3.3}$$

Recall, that F_θ is slow, as it performs numerical integration across the ray $r = (o, d)$.

Second, we employ the newly obtained F_{θ^*} as the teacher model for our DyLiN student model G_ϕ via KD. Specifically, we minimize the MSE loss against the respective pseudo ground truth ray color generated by F_{θ^*} across S ray samples:

$$\min_{\phi} \mathbb{E}_{t,r=(o,d)} \left[\| G_\phi(o,d,t) - F_{\theta^*}(o,d,t) \|_2^2 \right], \tag{3.4}$$

yielding $G_{\tilde{\phi}}$. Note how this is considerably different from R2L [65], which uses a static LFN that is distilled from a static NeRF.

Finally, we initialize our student model G_ϕ with parameters $\tilde{\phi}$ and fine-tune it using the original real video data:

$$\min_{\phi,\, \phi_0 = \tilde{\phi}} \mathbb{E}_{t,r=(o,d)} \left[\| G_\phi(o,d,t) - I(o,d,t) \|_2^2 \right], \tag{3.5}$$

obtaining ϕ^*.

3.4 Experimental Setup

3.4.1 Datasets

To test our hypotheses, we performed experiments on three types of dynamic scenes: synthetic, real and real controllable.

Synthetic Scenes. We utilized the synthetic 360° dynamic dataset introduced by [53], which contains 8 animated objects with complicated geometry and realistic non-Lambertian materials. Each dynamic scene consists of 50 to 200 training images and 20 testing images. We used 400×400 image resolution. We applied D-NeRF [53] as our teacher model with the publicly available pretrained weights.

Real Scenes. We collected real dynamic data from 2 sources. First, we utilized 5 topologically varying scenes provided by [51] (Broom, 3D Printer, Chicken, Americano and Banana), which were captured by a rig encompassing a pole with two Google Pixel 3 phones rigidly attached roughly 16 cm apart. Second, we collected human facial videos using an iPhone 13 Pro camera. We rendered both sets at 960×540 image resolution. We pretrained a HyperNeRF [51] teacher model from scratch for each scene.

3.4.2 Settings

Throughout our experiments, we use the settings listed below, many of which follow [65].

In order to retain efficiency, we define T_ω and H_ψ to be small MLPs, with T_ω consisting of 7 layers of 128 units with $r' \in \mathbb{R}^6$, and H_ψ having 6 layers of 64 units with $w \in \mathbb{R}^8$. Then, we use $K = 16$ sampled points to represent rays, where sampling is done randomly during training and evenly spaced during inference.

Contrary to T_ω and H_ψ, our LFN R_π is a very deep residual color MLP regressor, containing 88 layers with 256 units per layer, in order to have enough capacity to learn the video generation process.

We generate rays within eqs. (3.3) to (3.5) and (4.3) by sampling ray origins $o = (x_o, y_o, z_o)$ and normalized directions $d = (x_d, y_d, z_d)$ randomly from the uniform

distribution U as follows:

$$x_o \sim U(x_o^{min}, x_o^{max}), \quad x_d \sim U(x_d^{min}, x_d^{max}), \tag{3.6}$$
$$y_o \sim U(y_o^{min}, y_o^{max}), \quad y_d \sim U(y_d^{min}, y_d^{max}), \tag{3.7}$$
$$z_o \sim U(z_o^{min}, z_o^{max}), \quad z_d \sim U(z_d^{min}, z_d^{max}), \tag{3.8}$$

where the min, max bounds of the 6 intervals are inferred from the original training video. In addition to uniform sampling, we also apply the hard example mining strategy suggested in [65] to focus on fine-grained details. We used $S = 10\,000$ training samples during KD in (3.4).

Subsequently, we also randomly sample time step t uniformly from the unit interval: $t \sim U(0, 1)$.

During training, we used Adam [29] with learning rate 5×10^{-4} and batch size 4096.

We performed all experiments on single NVIDIA A100 GPUs.

3.4.3 Baseline Models

For testing our methods, we compared quality and speed against several baseline models, including NeRF [43], NV [38], NSFF [33], Nerfies [50], HyperNeRF [51], two variants of TiNeuVox [19], DirectVoxGo [60], Plenoxels [20], T-NeRF and D-NeRF [53], as well as CoNeRF [27].

In addition, we performed an ablation study by comparing against 2 simplified versions of our DyLiN architecture. First, we omitted both of our deformation and hyperspace MLPs and simply concatenated the time step t to the sampled ray points (essentially resulting in a dynamic R2L). This method is illustrated in fig. 3.2a. Second, we employed a pointwise deformation MLP (5 layers of 256 units) inspired by [53], which deforms points along a ray by predicting their offsets jointly, i.e., it can bend rays. This is contrast to our DyLiN, which deforms rays explicitly without bending and also applies a hyperspace MLP. This scheme is depicted in fig. 3.2b. In both baselines, the deep residual color MLP regressors were kept intact. Next, we also tested the effects of our fine-tuning procedure from (3.5) by training all of our models both with and without it. Lastly, we assessed the dependences on the number of

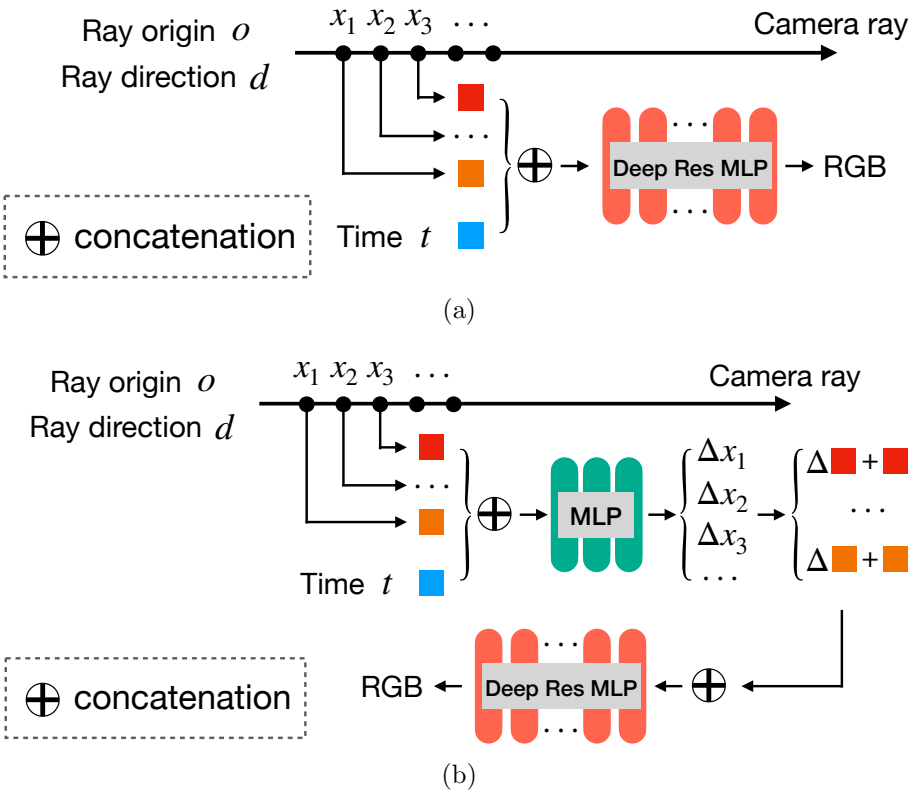


Figure 3.2: Our two ablated baseline models, omitting components of our DyLiN. (a) Without our two proposed MLPs. (b) Pointwise deformation MLP only, predicting offsets jointly.

sampled points along rays K and on the number of training samples S during KD in (3.4).

3.4.4 Evaluation Metrics

For quantitatively evaluating the quality of generated images, we calculated the Peak Signal-to-Noise Ratio (PSNR) [24] in decibels (dB), the Structural Similarity Index (SSIM) [48, 70], the Multi-Scale SSIM (MS-SSIM) [69] and the Learned Perceptual Image Patch Similarity (LPIPS) [76] metrics. Intuitively, PSNR is a pixelwise score, while SSIM and MS-SSIM also take pixel correlations and multiple scales into account, respectively, yet all of these tend to favor blurred images. LPIPS compares deep neural representations of images and is much closer to human perception, promoting semantically better and sharper images.

Furthermore, for testing space and time complexity, we computed the storage size of parameters in megabytes (MB) and measured the wall-clock time in milliseconds (ms) while rendering the synthetic Lego scene with each model.

3.5 Results

3.5.1 Quantitative Results

table 3.1 and table 3.2 contain our quantitative results for reconstruction quality on synthetic and real dynamic scenes, accordingly. We found that among prior works, TiNeuVox-B performed the best on synthetic scenes with respect to each metric. On real scenes, however, NSFF took the lead. Despite having strong metrics, NSFF is qualitatively poor and slow. Surprisingly, during ablation, even our most basic model (DyLiN without the two MLPs from fig. 3.2a) could generate perceptually better looking images than TiNeuVox-B, thanks to the increased training dataset size via KD. Incorporating the MLPs T_ω and H_ψ into the model each improved results slightly. Interestingly, fine-tuning on real data as in (3.5) gave a substantial boost. In addition, our relative PSNR improvement over the teacher model (table 3.1=+1.93 dB, up to +3.16 dB per scene; table 3.2=+2.7 dB, up to +13.14 dB) is better than that of R2L [65] (+1.4 dB, up to +2.8 dB).

Table 3.1: Quantitative results on synthetic dynamic scenes. Notations: Multi-Layer Perceptron (MLP), PD (pointwise deformation), FT (fine-tuning). We utilized D-NeRF as the teacher model for our DyLiNs. The winning numbers are highlighted in bold.

Method	PSNR↑	SSIM↑	LPIPS↓
NeRF[43]	19.00	0.8700	0.1825
DirectVoxGo[60]	18.61	0.8538	0.1688
Plenoxels[20]	20.24	0.8688	0.1600
T-NeRF[53]	29.51	0.9513	0.0788
D-NeRF[53]	30.50	0.9525	0.0663
TiNeuVox-S[19]	30.75	0.9550	0.0663
TiNeuVox-B[19]	**32.67**	0.9725	0.0425
DyLiN, w/o two MLPs, w/o FT (ours)	31.16	0.9931	0.0281
DyLiN, w/o two MLPs (ours)	32.07	0.9937	0.0196
DyLiN, PD MLP only, w/o FT (ours)	31.26	0.9932	0.0279
DyLiN, PD MLP only (ours)	31.24	0.9940	0.0189
DyLiN, w/o FT (ours)	31.37	0.9933	0.0275
DyLiN (ours)	32.43	**0.9943**	**0.0184**

Table 3.2: Quantitative results on real dynamic scenes. Notations: Multi-Layer Perceptron (MLP), PD (pointwise deformation), FT (fine-tuning). We utilized HyperNeRF as the teacher model for our DyLiNs. The winning numbers are highlighted in bold.

Method	PSNR↑	MS-SSIM↑
NeRF[43]	20.1	0.745
NV[38]	16.9	0.571
NSFF[33]	**26.3**	**0.916**
Nerfies[50]	22.2	0.803
HyperNeRF[51]	22.4	0.814
TiNeuVox-S[19]	23.4	0.813
TiNeuVox-B[19]	24.3	0.837
DyLiN, w/o two MLPs, w/o FT (ours)	23.8	0.882
DyLiN, w/o two MLPs (ours)	24.2	0.894
DyLiN, PD MLP only, w/o FT (ours)	23.9	0.885
DyLiN, PD MLP only (ours)	24.6	0.903
DyLiN, w/o FT (ours)	24.0	0.886
DyLiN (ours)	25.1	0.910

Table 3.3: Quantitative results for space and time complexity on the synthetic Lego scene. Notations: Multi-Layer Perceptron (MLP), PD (pointwise deformation), FT (fine-tuning).

Method	Storage (MB)	Wall-clock time (ms)
NeRF[43]	5.00	2,950.0
DirectVoxGo[60]	205.00	1,090.0
Plenoxels[20]	717.00	50.0
NV[38]	439.00	74.9
D-NeRF[53]	4.00	8,150.0
NSFF[33]	14.17	5,450.0
HyperNeRF[51]	15.36	2,900.0
TiNeuVox-S[19]	23.70	3,280.0
TiNeuVox-B[19]	23.70	6,920.0
DyLiN, w/o two MLPs, w/o FT (ours)	68.04	115.4
DyLiN, w/o two MLPs (ours)	68.04	115.4
DyLiN, PD MLP only, w/o FT (ours)	72.60	115.7
DyLiN, PD MLP only (ours)	72.60	115.7
DyLiN, w/o FT (ours)	70.11	116.0
DyLiN (ours)	70.11	116.0

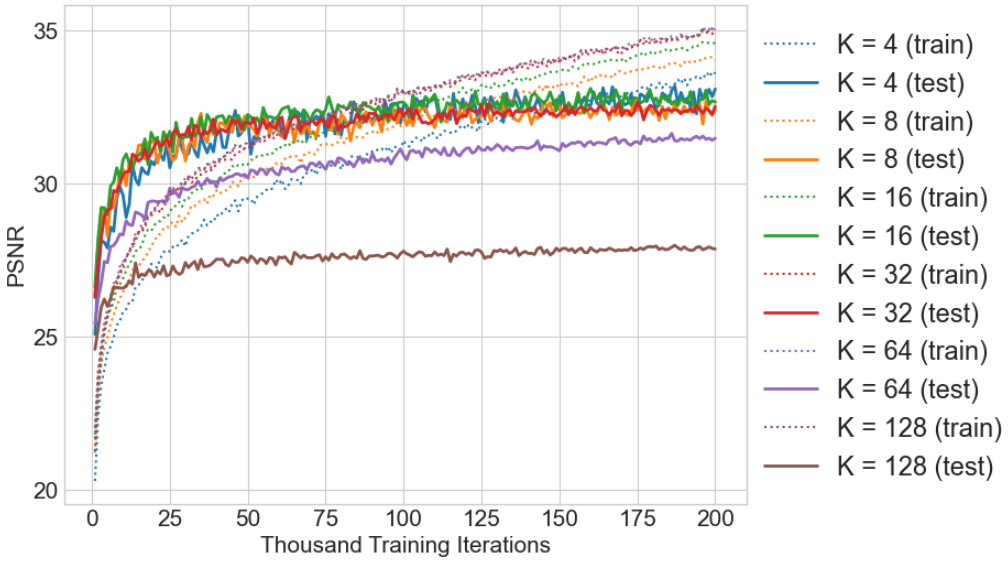
table 3.3 shows quantitative results for space and time complexity on the synthetic Lego scene. We found that there is a trade-off between the two metrics, as prior works are typically optimized for just one of those. In contrast, all of our proposed DyLiN variants settle at the golden mean between the two extremes. When compared to the strongest baseline TiNeuVox-B, our method requires 3 times as much storage but is nearly 2 orders of magnitude faster. Plenoxels and NV, the only methods that require less computation than ours, perform much worse in quality.

fig. 3.4 reports quantitative ablation results for dependencies on the number of sampled points per ray K and on the number of training samples during KD S, performed on the synthetic Standup scene. For dependence on K (fig. 3.4a), we found that there were no significant differences between test set PNSR scores for $K \in \{4, 8, 16, 32\}$, while we encountered overfitting for $K \in \{64, 128\}$. This justified our choice of $K = 16$ for the rest of our experiments. Regarding the effect of S (fig. 3.4b), overfitting occured for smaller sample sizes including $S \in \{100; 500; 1000; 5000\}$. The test and training set PSNR scores were much closer for $S = 10\,000$, validating our general setting.

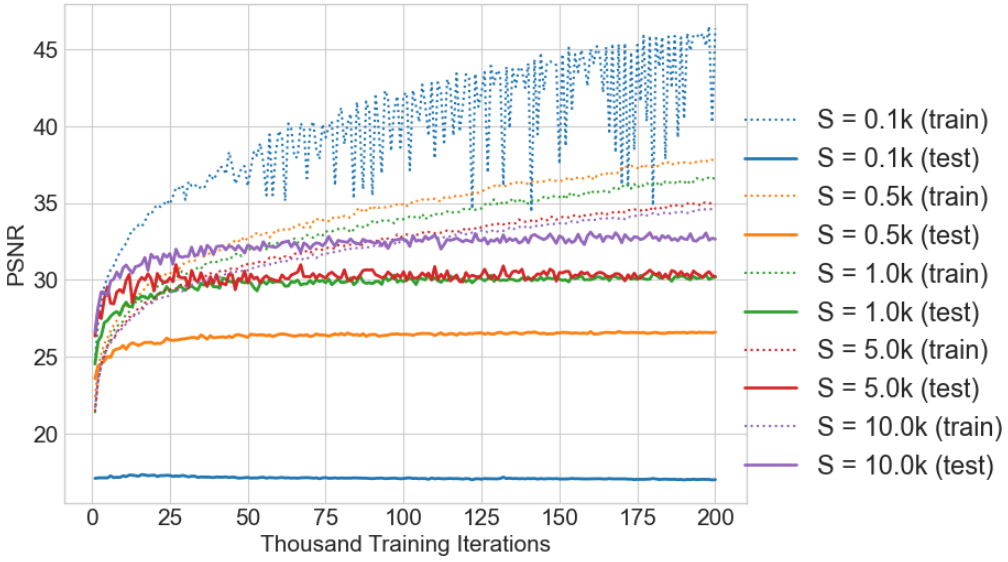
3.5.2 Qualitative Results

fig. A.10 and fig. 3.6 depict qualitative results for reconstruction quality on synthetic and real dynamic scenes, respectively. Both show that our full DyLiN model generated the sharpest, most detailed images, as it was able to capture cloth wrinkles (fig. A.9l) and the eye of the chicken (fig. 3.6f). The competing methods tended to oversmooth these features. We also ablated the effect of omitting fine-tuning (fig. A.9k, fig. 3.6e), and results declined considerably.

For the sake of completeness, fig. 3.7 illustrates qualitative ablation results for our model components on real dynamic scenes. We found that sequentially adding our two proposed MLPs T_ω and H_ψ improves the reconstruction, e.g., the gum between the teeth (fig. 3.7f) and the fingers (fig. 3.7l) become more and more apparent. Without the MLPs, these parts were heavily blurred (fig. 3.7d, fig. 3.7j).

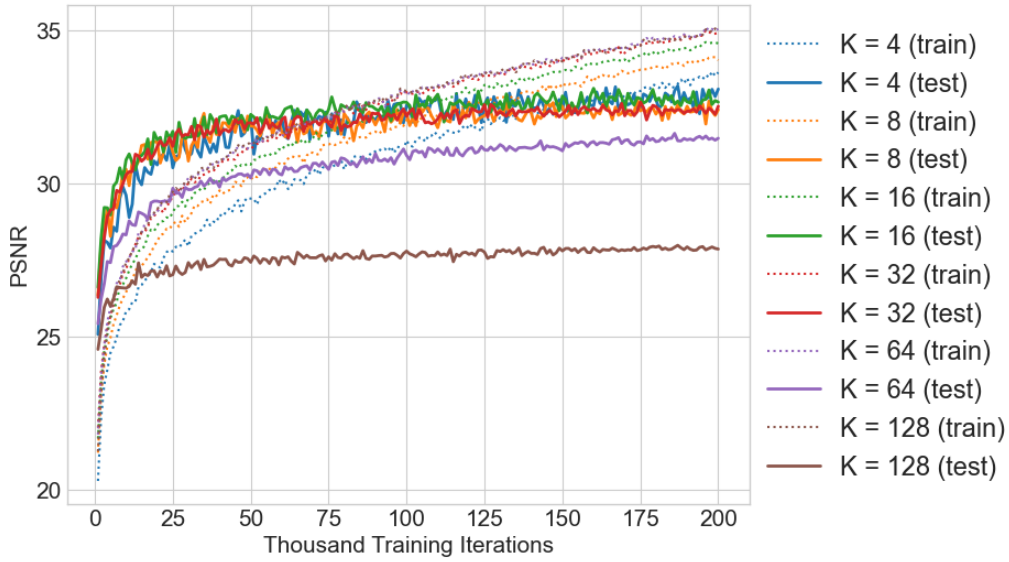


(a)

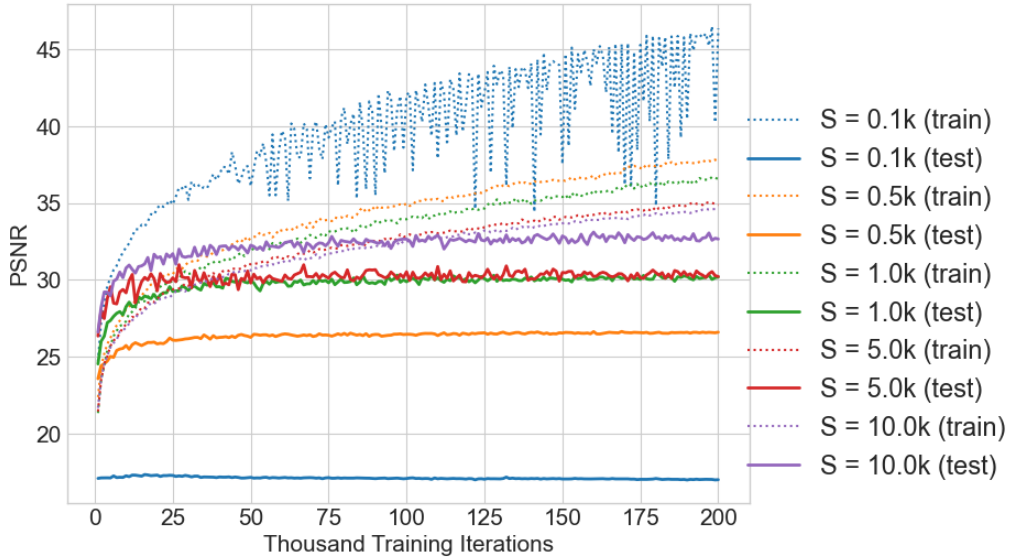


(b)

Figure 3.3: Quantitative results for ablation on the synthetic Standup scene. (a) Dependence on the number of sampled points K across ray r'. (b) Dependence on the number of training samples S during Knowledge Distillation (KD).

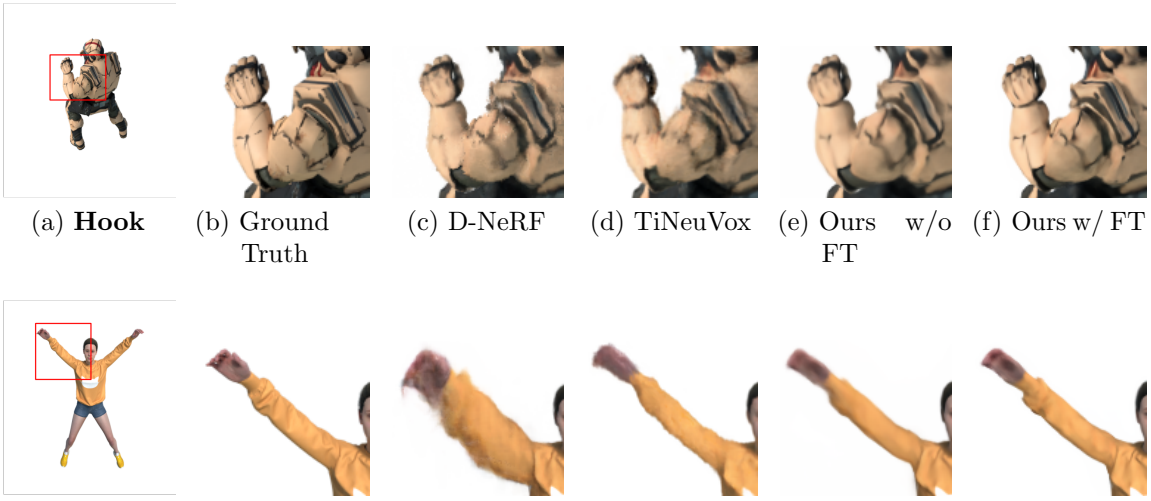


(a)



(b)

Figure 3.4: Quantitative results for ablation on the synthetic Standup scene. (a) Dependence on the number of sampled points K across ray r'. (b) Dependence on the number of training samples S during Knowledge Distillation (KD).



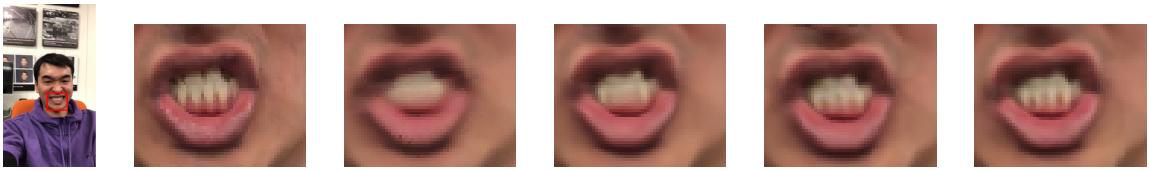
(a) **Hook** (b) Ground Truth (c) D-NeRF (d) TiNeuVox (e) Ours w/o FT (f) Ours w/ FT

(g) **Jumping Jacks** (h) Ground Truth (i) D-NeRF (j) TiNeuVox (k) Ours w/o FT (l) Ours w/ FT

Figure 3.5: Qualitative results on synthetic dynamic scenes. We compare our DyLiN (Ours-1, Ours-2) with the ground truth, the D-NeRF teacher model and TiNeuVox. Ours-1 and Ours-2 were trained without and with fine-tuning on the original data, respectively.

(a) **Chicken** (b) Ground Truth (c) HyperNeRF (d) TiNeuVox (e) Ours w/o FT (f) Ours w/ FT

Figure 3.6: Qualitative results on a real dynamic scene. We compare our DyLiN (Ours-1, Ours-2) with the ground truth, the HyperNeRF teacher model, and TiNeuVox. Ours-1 and Ours-2 were trained without and with fine-tuning on the original data, respectively.



(a) **Expression** (b) Ground Truth (c) HyperNeRF (d) Ours-1 (e) Ours-2 (f) Ours-3

(g) **Peel Banana** (h) Ground Truth (i) HyperNeRF (j) Ours-1 (k) Ours-2 (l) Ours-3

Figure 3.7: Qualitative results for ablation on real dynamic scenes. We compare our DyLiN (Ours-1, Ours-2, Ours-3) with the ground truth and the HyperNeRF teacher model. Ours-1 was trained without our two MLPs. Ours-2 was trained with the pointwise deformation MLP only. Ours-3 is our full model with both of our proposed MLPs.

3.6 Conclusion

We proposed the architecture for extending LFNs to dynamic scenes. Specifically, we introduced DyLiN, which models ray deformations without bending and lifts whole rays into a hyperspace. We trained our model via knowledge distillation from various dynamic NeRF teacher models. We found that DyLiN produces state-of-the-art quality even without ray bending, while is nearly 2 orders of magnitude faster than their strongest baselines.

Our method does not come without limitations, however. Most importantly, it focuses on speeding up inference, as it requires a pretrained teacher model, which can be expensive to obtain. In some experiments, our solution was outperformed in terms of the PSNR score. Using the winners as teacher models could improve performance. Additionally, distillation from multiple teacher models or joint training of the teacher and student models is also yet to be explored. Moreover, we currently represent rays implicitly by sampling K points along them, but increasing this number can lead to overfitting. An explicit ray representation may be more effective. Finally, voxelizing and quantizing our models could improve efficiency.

Our results are encouraging steps towards achieving real-time volumetric rendering and animation, and we hope that our work will contribute to the progress in these areas.

Chapter 4

Real-time Controllable Neural Face Avatars

In this chapter, we will delve into the construction of the real-time controllable neural face avatar system, building upon the findings and advancements presented in the previous chapters. Specifically, we will explore how the controllability discussed in Chapter 2 and the real-time capabilities outlined in Chapter 3 are integrated to form a cohesive and functional system.

4.1 Methods

In this section, we introduce CoDyLiN, which further augments DyLiN with controllability, via lifting attribute inputs to hyperspace with MLPs, and masking their hyperspace codes for disentanglement. In this case, we also train via KD, but the teacher model is a pretrained controllable NeRF.

4.1.1 Network Architecture

We demonstrate here that our DyLiN architecture from section 3.3.1 can be extended to the controllable scenario using attribute inputs with hyperspace MLPs and attention masks. Our proposed CoDyLiN network Q_τ is depicted in fig. 4.1.

Specifically, we start from DyLiN G_ϕ and add scalar inputs $\alpha_i \in [-1, 1]$, $i =$

$1, \ldots, n$ next to o, d, t. Intuitively, these are given strength values for specific local attributes, which can be interpolated continuously. n is the total number of attributes.

Each α_i is then processed independently with its own hyperspace MLP H_{i,ψ_i} to yield the hyperspace code w_i:

$$w_i = H_{i,\psi_i}(o, d, t). \tag{4.1}$$

Next, we include mask MLP regressors M_{i,ρ_i} to generate scalar attention masks $\hat{m}_i \in [0, 1]$ for each w_i (including $w_0 = w$):

$$\begin{aligned} \hat{m}_i &= M_{i,\rho_i}(w_i, w, o, d), \\ \hat{m}_0 &= 1 - \sum_{i=1}^{n} \hat{m}_i, \\ w'_i &= \hat{m}_i \cdot w_i, \quad i = 0, \ldots, n, \end{aligned} \tag{4.2}$$

This helps the architecture to spatially disentangle (i.e., localize) the effects of attributes α_i, while $\hat{m}_0$ can be understood as the space not affected by any attributes.

Finally, we sample K points on the ray similarly to section 3.3.1, concatenate those with the w'_i vectors, and process the result further with LFN R_π. Again, we can use a shorthand for the parameters: $\tau = [\omega, \psi, \psi_1, \ldots, \psi_n, \rho_1, \ldots, \rho_n, \pi]$.

Observe that without our MLPs H_{i,ψ_i}, M_{i,ρ_i}, $i = 1, \ldots, n$, our CoDyLiN reverts to our simpler DyLiN. Different from CoNeRF [27], we process rays instead of points, and use the α_i as inputs instead of targets.

4.1.2 Training Procedure

Akin to section 3.3.2, we split training into pretraining and distillation steps, but omit fine-tuning.

First, we pretrain a CoNeRF model E_ν [27] by randomly sampling (t, r, i), against 3 ground truths: ray color, attribute values α_i and 2D per-attribute masks $m_{2D,i}$. This yields us E_{ν^*}. For brevity, we omit the details of this step, and kindly forward the reader to Section 3 in [27].

Second, we distill from our teacher CoNeRF model E_{ν^*} into our student CoDyLiN Q_τ by randomly sampling $t, r, \alpha_1, \ldots, \alpha_n$, and minimizing the MSE against 2 pseudo

ground truths, i.e., ray colors and 2D masks $\bar{m}_{2D,i}$:

$$\min_{\tau} \mathbb{E}_{t,r=(o,d)} \Bigg[\|Q_\tau(o,d,t,\alpha_{1:n}) - \bar{E}_{\nu^*}(o,d,t,\alpha_{1:n})\|_2^2 + \lambda_m \cdot \sum_{i=0}^{n} \|\hat{m}_i(o,d,t,\alpha_i) - \bar{m}_{2D}(o,d,t,\alpha_{1:n})_i\|_2^2 \Bigg], \quad (4.3)$$

where $\bar{E}_\nu$ is identical to E_ν except for taking $\alpha_{1:n} = [\alpha_1, \ldots, \alpha_n]$ as input and outputting the masks $\bar{m}_{2D,i}$, $i = 0, \ldots, n$. We denote the result of the optimization as Q_{τ^*}.

We highlight that our teacher and student models are both controllable in this setup.

4.2 Experimental Setup

Real Controllable Scenes. We borrowed several real controllable scenes from [27] and our CoNFies (closing/opening eyes/mouth, and transformer), which are captured either with a Google Pixel 3a or an Apple iPhone 13 Pro, and contain annotations over various attributes. We applied image resolution of 480×270 pixels. We pretrained a CoNeRF [27] teacher model from scratch per scene.

4.2.1 Settings

The experimental setup for our CoDyLiN experiments closely follows that of DyLiN, with a slight modification. In our CoDyLiN experiments, we define each H_{i,ψ_i} to be a small MLP having 5 layers of 128 units with $w_i \in \mathbb{R}^8$. During training, we uniformly sample attributes within $[-1, 1]$: $\alpha_i \sim U(-1, 1)$, and let $\lambda_m = 0.1$.

4.3 Results

4.3.1 Quantitative Results

Our controllable numerical results are collected in table 4.1. In short, our CoDyLiN was able to considerably outperform CoNeRF with respect to MS-SSIM and speed.

Table 4.1: Quantitative results on real controllable scenes. We utilized CoNeRF as the teacher model for our CoDyLiN. The winning numbers are highlighted in bold.

	Eyes/Mouth			Transformer		
Method	PSNR↑	MS-SSIM↑	Wall-clock time (ms)	PSNR↑	MS-SSIM↑	Wall-clock time (ms)
CoNeRF[27]	**21.4658**	0.7458	6230.0	23.0319	0.8878	4360.0
CoDyLiN (ours)	21.4655	**0.9510**	**116.3**	**23.5882**	**0.9779**	**116.0**

4.3.2 Qualitative Results

We present the visualization of our results in Figure 4.2, showcasing the effectiveness of CoDyLiN in achieving precise control over facial expressions. Notably, our system demonstrates remarkable speed, significantly outperforming previous methods. For a more comprehensive overview of our findings, additional results can be found in the supplementary material.

4.4 Conclusion

In conclusion, we introduce CoDyLiN, a pioneering real-time controllable neural face avatar system that combines the strengths of DyLiN and CoNFies. CoDyLiN inherits real-time capabilities from DyLiN, enabling fast and efficient rendering of facial animations, while incorporating controllability from CoNFies, granting users precise control over various attributes and expressions.

Our proposed system represents a significant step forward in the field of facial animation, providing a novel solution that achieves both real-time performance and fine-grained control. By integrating these key features, CoDyLiN offers a versatile platform for generating lifelike and expressive facial avatars in interactive applications, virtual communication, and entertainment.

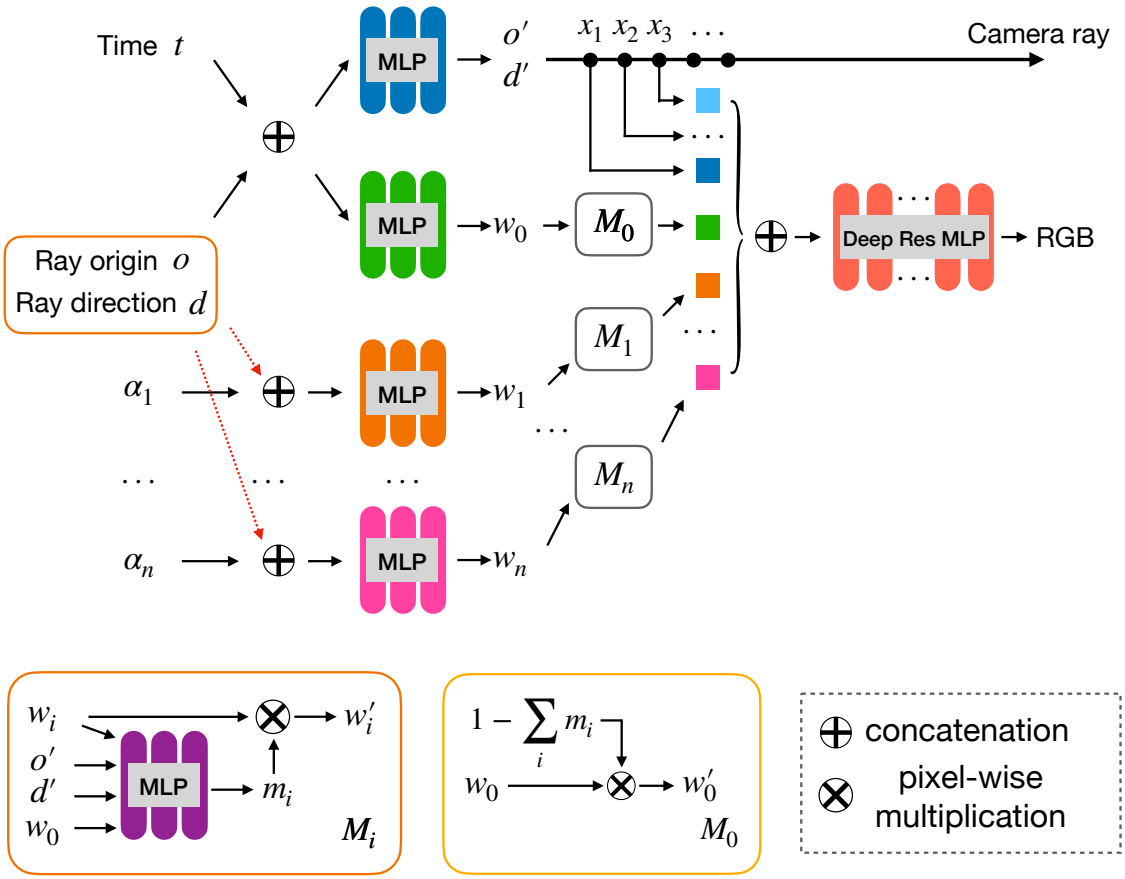


Figure 4.1: Schematic diagram of our proposed CoDyLiN architecture. We augment our DyLiN (blue, green, red) by introducing scalar attribute inputs $\alpha_i \in [-1, 1]$, $i = 1, \ldots, n$ and lifting them to their respective hyperspace codes w_i (orange, ..., pink MLPs). Next, M_i disentangles w_i from w_j, $j \neq i$ by masking it into w_i' (orange, ..., pink boxes and bottom insets). We concatenate the sampled points x_k, $k = 1, \ldots, K$ with the w_i', $i = 1, \ldots, n$ and predict the RGB color corresponding to the inputs (red). Arrows from (o', d') and w_0 to M_i are omitted from the top figure for simplicity. Compare this with fig. 3.1.

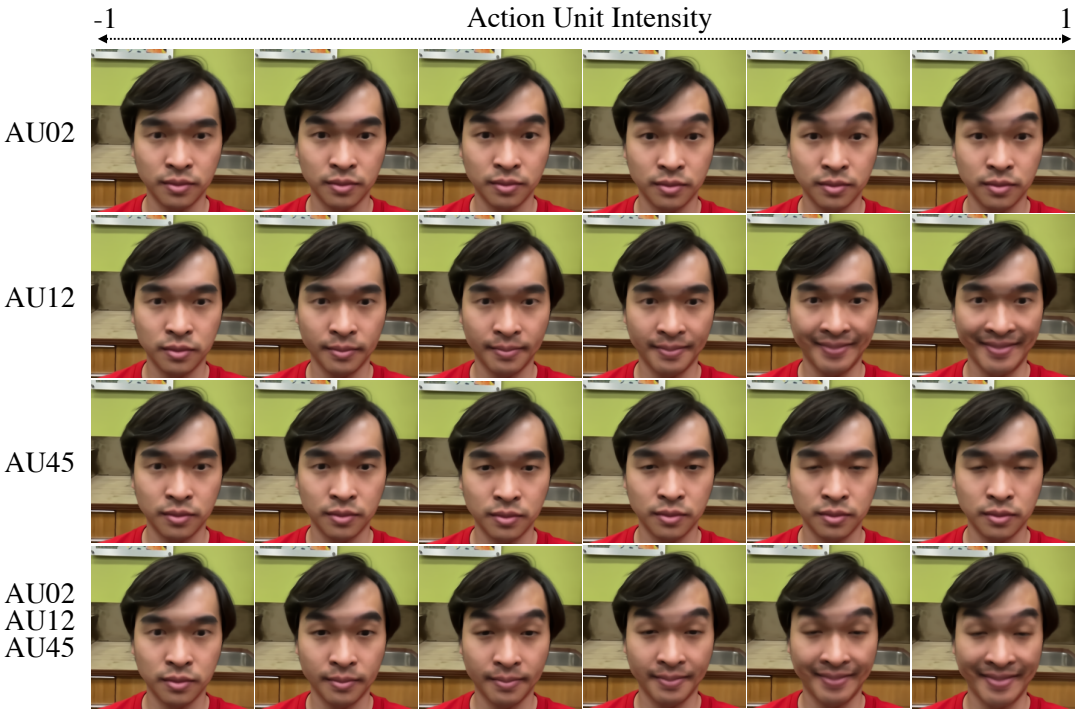


Figure 4.2: Control using single AU and their combinations using our CoDyLiN. AU02 is an outer brow raiser. AU12 is a lip corner puller. AU45 is blink.

Chapter 5

Conclusions

In conclusion, this thesis presents the proposal of a real-time controllable neural face avatar system, making notable contributions to the research area. The following contributions have been made:

Firstly, we achieved fully automatic annotation of facial data through the application of automated facial action recognition (AFAR). By characterizing facial expressions as a combination of action units (AU) and their intensities, we eliminated the need for labor-intensive manual annotation. This breakthrough significantly improves data processing efficiency and alleviates the burden of annotation tasks.

Secondly, we introduced an innovative neural representation that enables high-fidelity 3D reconstruction and precise control of intricate facial movements. Through a disentangled feature space, each region of the representation is independent, allowing for fine-grained control over specific aspects of facial expressions. This contribution advances the state-of-the-art in capturing realistic and expressive facial animations.

Furthermore, we implemented real-time rendering capabilities throughout the entire system by leveraging the concept of Light Field Networks (LFNs). This integration enables the generation of facial animations in real time, providing seamless and immediate visualization of rendered facial expressions. The achievement of real-time performance enhances the interactive and immersive nature of the user experience.

Collectively, these contributions pave the way for novel advancements in the realm of real-time controllable neural face avatars. By automating annotation processes,

improving the fidelity of 3D reconstruction, and achieving real-time performance, our research significantly enhances the quality and interactivity of virtual facial representations.

We anticipate that the proposed system and its contributions will foster progress in various fields, including virtual communication, entertainment, gaming, and human-computer interaction. This thesis sets the stage for further exploration and development in the exciting domain of real-time controllable neural face avatars, contributing to the advancement of technology and enriching the user experience in virtual environments.

Appendix A

Supplementary Material

A.1 Overview

In this supplementary material, we provide detailed quantitative and additional qualitative results, showcasing the benefits of our proposed CoNFies, DyLiN and CoDyLiN methods. Furthermore, we also provide the training times one should expect given our current setup.

A.2 CoNFies Architecture

We show the architectures of our CoNFies in this section: attribute mapping network A in Fig. A.1, spatial deformation field T in Fig. A.2, ambient slicing surface mapping network H in Fig. A.3, mask and uncertainty prediction network $M(B)$ in Fig. A.4, and rendering network in Fig. A.5. All networks only contain fully connected layers and all the hidden layers use ReLU as activation function. The numbers in each blocks indicate the kernel numbers used in corresponding layers. Our network architectures are similar to CoNeRF except that we introduce decoupling mask and uncertainty mechanism.

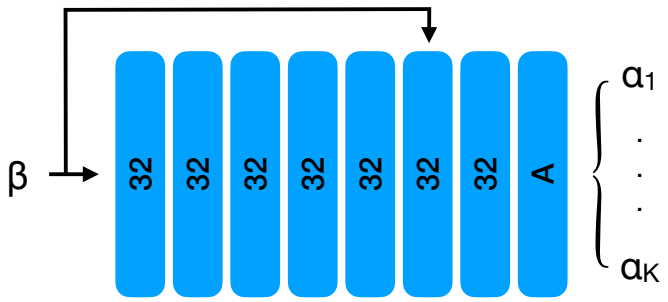


Figure A.1: Attribute mapping A takes a per-image learnable latent code β and outputs attributes $\alpha_{1...K}$. We use tanh as activation function to that the attributes have the range of (-1, 1).

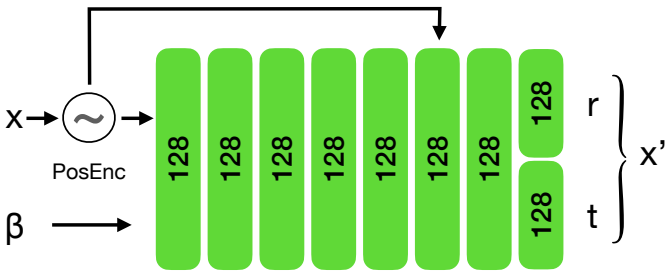


Figure A.2: Spatial deformation field T takes β and raw coordinates x to generate quaternion r as rotation and outputs translation t. Spatial deformation point x' is obtained through applying affine transform on x using r and t.

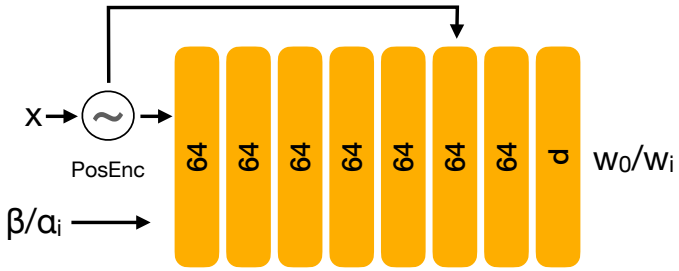


Figure A.3: Ambient slicing surface mapping network H takes β or attribute α_i along with raw coordinates x to learn the ambient space w_0 or w_i. w_0 is generated using β as input and w_i is generated using α_i as input.

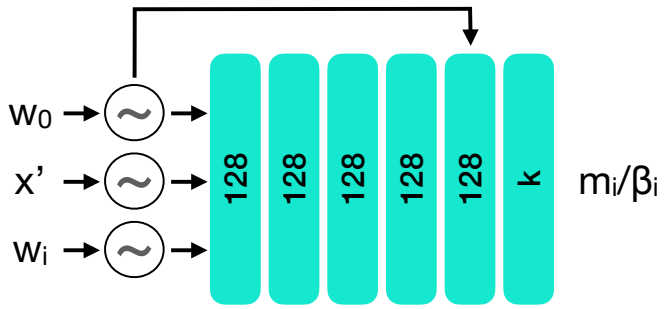


Figure A.4: Mask and uncertainty prediction network $M(B)$ has the same architecture, which takes spatial deformation point x', canonical ambient space w_0 and corresponding ambient space w_i. It's worth noting that each m_i can take multiple w_i as input and each β_i takes one w_i as input.

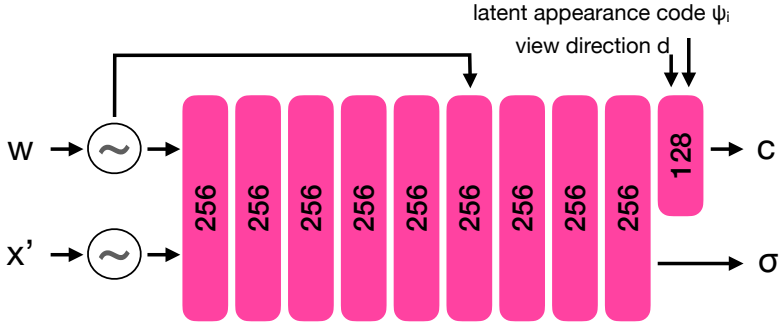


Figure A.5: Rendering network is the same as the template NeRF except the input dimension is adjusted according to spatial deformation point x' and ambient space w.

Table A.5: Per-scene quantitative results on synthetic dynamic scenes. Notations: Multi-Layer Perceptron (MLP), PD (pointwise deformation), FT (fine-tuning). We utilized D-NeRF as the teacher model for our DyLiNs. The winning numbers are highlighted in bold.

	Hell Warrior			Mutant			Hook			Bouncing Balls		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF[43]	13.52	0.8100	0.2500	20.31	0.9100	0.0900	16.65	0.8400	0.1900	20.26	0.9100	0.2000
DirectVoxGo[60]	13.51	0.7500	0.2500	19.45	0.8900	0.1200	16.16	0.8000	0.2100	20.20	0.8700	0.2200
Plenoxels[20]	15.19	0.7800	0.2700	21.44	0.9100	0.0900	17.90	0.8100	0.2100	21.30	0.8900	0.1800
T-NeRF[53]	23.19	0.9300	0.0800	30.56	0.9600	0.0400	27.21	0.9400	0.0600	37.81	0.9800	0.1200
D-NeRF[53]	25.10	0.9500	0.0600	31.29	0.9700	0.0200	29.25	0.9600	0.1100	38.93	0.9800	0.1000
TiNeuVox-S[19]	27.00	0.9500	0.0900	31.09	0.9600	0.0500	29.30	0.9500	0.0700	39.05	0.9900	0.0600
TiNeuVox-B[19]	**28.17**	0.9700	0.0700	33.61	0.9800	0.0300	**31.45**	0.9700	0.0500	40.73	0.9900	0.0400
DyLiN, w/o two MLPs, w/o FT (ours)	26.81	0.9885	0.0363	32.13	0.9961	0.0186	29.89	0.9922	0.0297	39.78	0.9997	0.0099
DyLiN, w/o two MLPs (ours)	27.73	0.9893	0.0317	33.26	0.9971	0.0101	30.20	0.9928	0.0187	41.13	**0.9998**	0.0064
DyLiN, PD MLP only, w/o FT (ours)	26.82	0.9886	0.0362	32.13	0.9963	0.0185	29.94	0.9923	0.0296	39.70	0.9996	0.0096
DyLiN, PD MLP only (ours)	27.75	0.9896	0.0302	33.47	0.9972	0.0102	30.39	0.9930	**0.0186**	41.52	**0.9998**	**0.0062**
DyLiN, w/o FT (ours)	26.90	0.9887	0.0360	32.17	0.9963	0.0182	29.99	0.9923	0.0289	40.02	0.9997	0.0098
DyLiN (ours)	27.79	**0.9898**	**0.0298**	**33.80**	**0.9974**	**0.0086**	30.49	**0.9931**	**0.0186**	**41.59**	**0.9998**	**0.0062**

	Lego			T-Rex			Stand Up			Jumping Jacks		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF[43]	20.30	0.7900	0.2300	24.29	0.9300	0.1300	18.19	0.8900	0.1400	18.28	0.8800	0.2300
DirectVoxGo[60]	21.13	0.9000	0.1000	23.27	0.9200	0.0900	17.58	0.8600	0.1600	17.80	0.8400	0.2000
Plenoxels[20]	21.97	0.9000	0.1100	25.18	0.9300	0.0800	18.76	0.8700	0.1500	20.18	0.8600	0.1900
T-NeRF[53]	23.82	0.9000	0.1500	30.19	0.9600	0.1300	31.24	0.9700	0.0200	32.01	0.9700	0.0300
D-NeRF[53]	21.64	0.8300	0.1600	31.75	0.9700	0.0300	32.79	0.9800	0.0200	32.80	0.9800	0.0300
TiNeuVox-S[19]	24.35	0.8800	0.1300	29.95	0.9600	0.0600	32.89	0.9800	0.0300	32.33	0.9700	0.0400
TiNeuVox-B[19]	**25.02**	0.9200	0.0700	32.70	0.9800	0.0300	35.43	0.9900	0.0200	**34.23**	0.9800	0.0300
DyLiN, w/o two MLPs, w/o FT (ours)	22.11	0.9747	0.0612	31.35	0.9978	0.0290	33.98	0.9973	0.0140	33.24	0.9981	0.0260
DyLiN, w/o two MLPs (ours)	22.42	0.9761	0.0493	32.80	0.9984	0.0170	35.31	0.9980	0.0084	33.67	0.9984	0.0155
DyLiN, PD MLP only, w/o FT (ours)	22.13	0.9748	0.0618	32.18	0.9982	0.0282	33.97	0.9973	0.0140	33.19	0.9982	0.0257
DyLiN, PD MLP only (ours)	22.76	0.9775	0.0452	32.77	**0.9985**	0.0176	35.56	0.9981	0.0082	33.68	0.9984	0.0152
DyLiN, w/o FT (ours)	22.24	0.9754	0.0600	32.24	0.9982	0.0276	34.15	0.9974	0.0141	33.23	0.9983	0.0256
DyLiN (ours)	23.10	**0.9791**	**0.0443**	**32.91**	**0.9985**	**0.0168**	**35.95**	**0.9983**	**0.0074**	33.84	**0.9985**	**0.0151**

A.3 DyLiN Per-Scene Quantitative Results

For the sake of completeness, we provide the detailed per-scene quantitative results for reconstruction quality (PSNR, SSIM, MS-SSIM, LPIPS) on the synthetic (table A.5) and real (table A.6) dynamic scenes, extending Tab. 1 and Tab. 2 in the main paper that average these numbers across the scenes. Accordingly, we found that our DyLiN performs the best with respect to the SSIM and LPIPS metrics, generating perceptually better images, yet it sometimes falls behind in terms of PSNR and MS-SSIM that may prefer blurred results. Knowledge distillation improves a lot, our deformation and hyperspace MLPs yield slightly better results, while fine-tuning on the original training data gives a considerable boost.

Table A.6: Per-scene quantitative results on real dynamic scenes. Notations: Multi-Layer Perceptron (MLP), PD (pointwise deformation), FT (fine-tuning), N/A (not available in the cited research paper). We utilized HyperNeRF as the teacher model for our DyLiNs. The winning numbers are highlighted in bold.

	Broom		3D Printer		Chicken	
Method	PSNR↑	MS-SSIM↑	PSNR↑	MS-SSIM↑	PSNR↑	MS-SSIM↑
NeRF[43]	19.90	0.653	20.70	0.780	19.90	0.777
NV [38]	17.70	0.623	16.20	0.665	17.60	0.615
NSFF [33]	**26.10**	**0.871**	**27.70**	**0.947**	26.90	0.944
Nerfies [50]	19.20	0.567	20.60	0.830	26.70	0.943
HyperNeRF [51]	19.30	0.591	20.00	0.821	26.90	0.948
TiNeuVox-S[19]	21.90	0.707	22.70	0.836	27.00	0.929
TiNeuVox-B[19]	21.50	0.686	22.80	0.841	**28.30**	0.947
DyLiN, w/o two MLPs, w/o FT (ours)	21.98	0.808	22.99	0.899	26.89	0.948
DyLiN, w/o two MLPs (ours)	22.04	0.811	23.16	0.905	27.35	0.954
DyLiN, PD MLP only, w/o FT (ours)	22.02	0.805	23.04	0.903	26.88	0.948
DyLiN, PD MLP only (ours)	22.14	0.815	23.19	0.906	27.53	0.955
DyLiN, w/o FT (ours)	22.04	0.809	23.06	0.902	26.91	0.948
DyLiN (ours)	22.14	0.823	23.21	0.906	27.62	**0.956**
	Peel Banana		Americano		Expressions	
Method	PSNR↑	MS-SSIM↑	PSNR↑	MS-SSIM↑	PSNR↑	MS-SSIM↑
NeRF[43]	20.00	0.769	N/A	N/A	N/A	N/A
NV [38]	15.90	0.380	N/A	N/A	N/A	N/A
NSFF [33]	24.60	0.902	N/A	N/A	N/A	N/A
Nerfies [50]	22.40	0.872	N/A	N/A	N/A	N/A
HyperNeRF [51]	23.30	0.896	18.42	0.720	25.40	0.958
TiNeuVox-S[19]	22.10	0.780	N/A	N/A	N/A	N/A
TiNeuVox-B[19]	24.40	0.873	N/A	N/A	N/A	N/A
DyLiN, w/o two MLPs, w/o FT (ours)	23.38	0.872	18.45	0.722	25.36	0.950
DyLiN, w/o two MLPs (ours)	24.35	0.906	30.85	0.977	26.33	0.967
DyLiN, PD MLP only, w/o FT (ours)	23.70	0.882	18.47	0.722	25.55	0.960
DyLiN, PD MLP only (ours)	25.72	0.936	31.01	0.978	26.33	0.967
DyLiN, w/o FT (ours)	23.97	0.886	18.48	0.722	26.51	0.969
DyLiN (ours)	**27.36**	**0.952**	**31.56**	**0.982**	**26.91**	**0.974**

A.4 More Qualitative Results of DyLiN

We provide additional qualitative results for 3 experiments.

First, fig. A.10 depicts more qualitative results for reconstruction quality on synthetic dynamic scenes, extending Fig. 6 in the main paper. Specifically, the Standup scene includes buttons on the shirt of the avatar (fig. A.9b), and the baselines are all missing them (figs. A.9c and A.9d), whereas our full method is capable of reconstructing such details (fig. A.9f). Furthermore, the Bouncing Ball scene involves shadow casting (fig. A.9h). Inside the shadowed area, D-NeRF [53] produces horizontal artifacts (fig. A.9i), while TiNeuVox [19] predicts an inaccurate boundary (fig. A.9j). Again, our full model outputs the correct shadow (fig. A.9l).

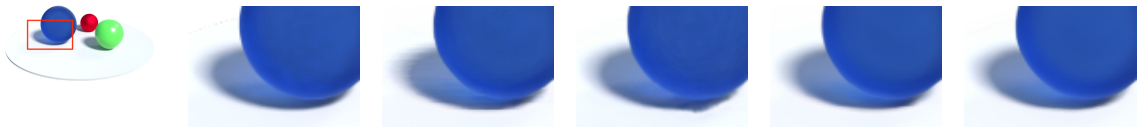
Second, fig. 3.7 shows qualitative results for ablation on the synthetic Standup scene using a D-NeRF teacher model, complementing Fig. 8 in the main paper that is restricted to real scenes and distilling from HyperNeRF [51]. D-NeRF gives an oversmoothed prediction (fig. A.10c), whereas the two MLPs of our DyLiN gradually reduce the blurriness of the face (figs. A.10d to A.10f).

Lastly, fig. A.11 illustrates qualitative results for the real controllable Transformer scene, complementing the numbers of Tab. 4 in the main paper. We portray the effects of altering the attribute input $\alpha_i \in [-1, 1]$, which encodes the body pose of the character. We found that the CoNeRF [27] teacher model produces yellow color artifacts outside the boundary of the character (see, e.g., top row 1^{st} inset), whereas our CoDyLiN student model captures the boundary well.

A.5 Training Times for DyLiN and CoDyLiN

On a single NVIDIA A100 GPU, the full process takes $\approx$ 38–43 h, including 5–10 h to train the teacher, 13 h for drawing $S = 10\,000$ training samples for KD, and 20 h for training the student via KD.

(a) **Standup** (b) Ground Truth (c) D-NeRF (d) TiNeuVox (e) Ours-1 (f) Ours-2



(g) **Bouncing Ball** (h) Ground Truth (i) D-NeRF (j) TiNeuVox (k) Ours-1 (l) Ours-2

Figure A.9: More qualitative results on synthetic dynamic scenes. We compare our DyLiN (Ours-1, Ours-2) with the ground truth, the D-NeRF teacher model, and TiNeuVox. Ours-1 and Ours-2 were trained without and with fine-tuning on the original data, respectively.

(a) **Standup** (b) Ground Truth (c) D-NeRF (d) Ours-1 (e) Ours-2 (f) Ours-3

Figure A.10: Qualitative results for ablation on the synthetic Standup scene. We compare our DyLiN (Ours-1, Ours-2, Ours-3) with the ground truth and the D-NeRF teacher model. Ours-1 was trained without our two MLPs. Ours-2 was trained with pointwise deformation MLP only. Ours-3 is our full model with both of our proposed two MLPs.

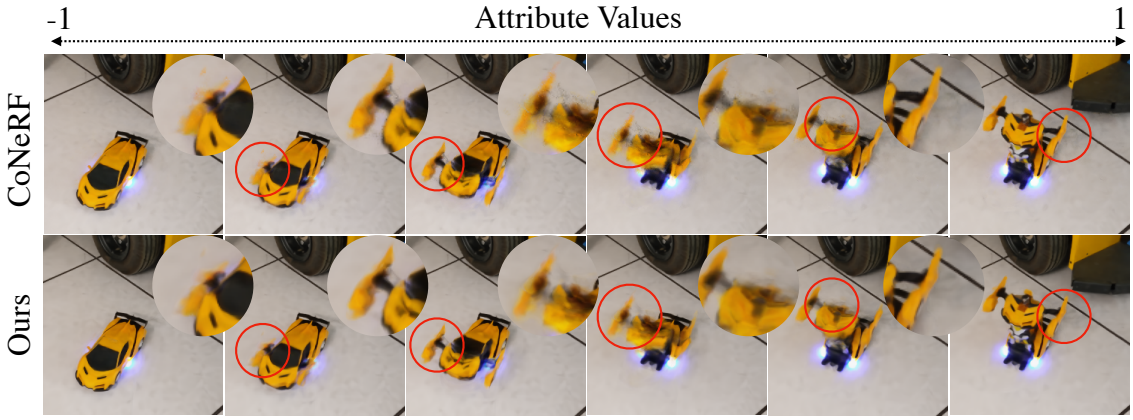


Figure A.11: Qualitative results on the real controllable Transformer scene. We utilized CoNeRF [27] as the teacher model for our CoDyLiN. Red circles indicate regions enlarged in insets. Best viewed zoomed in.

Bibliography

[1] E.H. Adelson and J.Y.A. Wang. Single Lens Stereo with a Plenoptic Camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):99–106, 1992. doi: 10.1109/34.121783. 3.2

[2] Zadeh Amir, Baltrusaitis Tadas, and Morency Louis-Philippe. Convolutional experts constrained local model for facial landmark detection. *Proceedings of the IEEE CVPRW*, pages 2051–2059, 2017. 2.3.1

[3] ShahRukh Athar, Zhixin Shu, and Dimitris Samaras. Flame-in-nerf: Neural control of radiance fields for free view face animation. *arXiv preprint arXiv:2108.04913*, 2021. 2.2.2

[4] ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. Rignerf: Fully controllable neural 3d portraits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20364–20373, 2022. 2.1, 2.2.2

[5] Tadas Baltrusaitis, Peter Robinson, and Louis-Philippe Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 354–361, 2013. 2.3.1

[6] Tadas Baltrušaitis, Marwa Mahmoud, and Peter Robinson. Cross-dataset learning and person-specific normalisation for automatic action unit detection. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 6, pages 1–6. IEEE, 2015. 2.3.1

[7] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 59–66. IEEE, 2018. 2.1, 2.3.1, 2.4.2

[8] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, et al. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *Proc. IEEE/CVF ICCV*, pages 5855–5864, 2021. 3.1

[9] James R Bergen and Edward H Adelson. The Plenoptic Function and the

Elements of Early Vision. *Comput. Model. Vis. Process.*, 1:8, 1991. 3.2

[10] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 2.4.1

[11] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model Compression. In *Proc. 12th ACM SIGKDD ICKDDM*, pages 535–541, 2006. 3.2

[12] Chen Cao, Tomas Simon, Jin Kyu Kim, Gabe Schwartz, Michael Zollhoefer, Shunsuke Saito, Stephen Lombardi, Shih-En Wei, Danielle Belko, Shoou-I Yu, Yaser Sheikh, and Jason Saragih. Authentic volumetric avatars from a phone scan. *ACM Transactions on Graphics (TOG)*, 2022. 2.2.2

[13] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 2.1

[14] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning Efficient Object Detection Models with Knowledge Distillation. *Adv. NeurIPS*, 30, 2017. 3.2

[15] Zhiqin Chen. *IM-NET: Learning implicit fields for generative shape modeling.* PhD thesis, Applied Sciences: School of Computing Science, 2019. 3.1

[16] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *Proc. IEEE/CVF CVPR*, pages 12882–12891, 2022. 3.1

[17] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5154–5163, 2020. 2.2.2

[18] Itir Onal Ertugrul, László A Jeni, Wanqiao Ding, and Jeffrey F Cohn. Afar: A deep learning based tool for automated facial affect recognition. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–1. IEEE, 2019. 2.1

[19] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, et al. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. *arXiv:2205.15285*, 2022. 3.1, 3.2, 3.4.3, 3.1, 3.2, 3.3, A.5, A.6, A.4

[20] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, et al. Plenoxels: Radiance Fields Without Neural Networks. In *Proc. IEEE/CVF CVPR*, pages 5501–5510, 2022. 2.1, 3.1, 3.4.3, 3.1, 3.3, A.5

[21] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. In *Proc.*

IEEE/CVF CVPR, pages 8649–8658, 2021. 2.1, 2.2.2, 3.1, 3.2

[22] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021. 2.2.1

[23] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The Lumigraph. In *Proc. 23rd CGIT*, pages 43–54, 1996. 3.2

[24] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *20th ICPR*, pages 2366–2369. IEEE, 2010. 3.4.4

[25] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958, 2021. 2.1

[26] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. 2.3.2

[27] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. CoNeRF: Controllable Neural Radiance Fields. In *Proc. IEEE/CVF CVPR*, pages 18623–18632, 2022. (document), 2.1, 2.2.2, 2.3.2, 2.3.2, 2.3.2, 2.3.2, 2.4.1, 2.4.2, 2.4.5, 3.1, 3.4.3, 4.1.1, 4.1.2, 4.2, 4.1, A.4, A.11

[28] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. 2.2.2

[29] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014. 2.4.1, 3.4.2

[30] Mohammad Rami Koujan, Michail Christos Doukas, Anastasios Roussos, and Stefanos Zafeiriou. Head2head: Video-based neural head synthesis. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, pages 16–23. IEEE, 2020. 2.2.2

[31] Marek Kowalski, Stephan J Garbin, Virginia Estellers, Tadas Baltrušaitis, Matthew Johnson, and Jamie Shotton. Config: Controllable neural face image generation. In *European Conference on Computer Vision*, pages 299–315. Springer, 2020. 2.2.2

[32] Marc Levoy and Pat Hanrahan. Light Field Rendering. In *Proc. 23rd CGIT*, pages 31–42, 1996. 3.2, 3.2

[33] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. In *Proc. IEEE/CVF CVPR*, 2021. 3.4.3, 3.2, 3.3, A.6

[34] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2.2.1

[35] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2.3.2

[36] David B Lindell, Julien NP Martel, and Gordon Wetzstein. AutoInt: Anutomatic Integration for Fast Neural Volume Rendering. In *Proc. IEEE/CVF CVPR*, pages 14556–14565, 2021. 3.1, 3.2

[37] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural Sparse Voxel Fields. *Adv. NeurIPS*, 33:15651–15663, 2020. 3.1, 3.2

[38] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, et al. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019. 3.4.3, 3.2, 3.3, A.6

[39] Shugao Ma, Tomas Simon, Jason Saragih, Dawei Wang, Yuecheng Li, Fernando De La Torre, and Yaser Sheikh. Pixel codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 64–73, 2021. 2.2.2

[40] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, et al. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proc. IEEE/CVF CVPR*, pages 7210–7219, 2021. 3.1

[41] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proc. IEEE/CVF CVPR*, pages 4460–4470, 2019. 3.1

[42] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 1, 2, 2.1, 2.3.2, 2.4.1, 2.4.5

[43] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, et al. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Commun. ACM*, 65(1):99–106, 2021. 3.1, 3.4.3, 3.1, 3.2, 3.3, A.5, A.6

[44] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images. In *Proc. IEEE/CVF CVPR*, pages 16190–16199, 2022. 3.1

[45] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 2.2.1

[46] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, et al. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. In *CGF*, volume 40, pages 45–59. Wiley Online Library, 2021. 3.1

[47] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021. 2.2.1

[48] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In *ICML*, pages 2642–2651. PMLR, 2017. 3.4.4

[49] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proc. IEEE/CVF CVPR*, pages 165–174, 2019. 2.3.2, 3.1

[50] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, et al. Nerfies: Deformable Neural Radiance Fields. In *Proc. IEEE/CVF ICCV*, pages 5865–5874, 2021. 2.2.1, 2.3.2, 2.4.1, 2.4.5, 3.1, 3.2, 3.3.1, 3.4.3, 3.2, A.6

[51] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, et al. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Trans. Graph.*, 40(6):1–12, 2021. 2.1, 2.2.1, 2.2.2, 2.3.2, 2.4.1, 2.4.5, 3.1, 3.2, 3.3.1, 3.3.2, 3.4.1, 3.4.3, 3.2, 3.3, A.6, A.4

[52] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. *https://arxiv.org/abs/2011.13961*, 2020. 2.1

[53] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proc. IEEE/CVF CVPR*, pages 10318–10327, 2021. 2.2.1, 3.1, 3.2, 3.3.2, 3.4.1, 3.4.3, 3.4.3, 3.1, 3.3, A.5, A.4

[54] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, et al. DeRF: Decomposed Radiance Fields. In *Proc. IEEE/CVF CVPR*, pages 14153–14161, 2021. 3.1

[55] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *Proc.*

IEEE/CVF ICCV, pages 14335–14345, 2021. 3.1, 3.2

[56] Erika L Rosenberg and Paul Ekman. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, 2020. 2.1

[57] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964. 2.3.1

[58] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020. 2.2.1

[59] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering. In *Proc. NeurIPS*, 2021. 1.3, 3.1, 3.2

[60] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *Proc. IEEE/CVF CVPR*, 2022. 3.4.3, 3.1, 3.3, A.5

[61] Ayush Tewari, Mohamed Elgharib, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Pie: Portrait image embedding for semantic control. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 2.2.2

[62] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6142–6151, 2020. 2.2.2

[63] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, et al. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video. In *Proc. IEEE/CVF ICCV*, pages 12959–12970, 2021. 2.1, 2.2.1, 3.1, 3.2

[64] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative Distillation for Ultra-Resolution Universal Style Transfer. In *Proc. IEEE/CVF CVPR*, pages 1860–1869, 2020. 3.2

[65] Huan Wang, Jian Ren, Zeng Huang, Kyle Olszewski, Menglei Chai, et al. R2L: Distilling Neural Radiance Field to Neural Light Field for Efficient Novel View Synthesis. *arXiv:2203.17261*, 2022. 3.1, 3.2, 3.2, 3.3.2, 3.4.2, 3.4.2, 3.5.1

[66] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, et al. Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-time. In

Proc. IEEE/CVF CVPR, pages 13524–13534, 2022. 3.1, 3.2

[67] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. 3.2

[68] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, et al. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *Adv. NeurIPS*, 34:27171–27183, 2021. 3.1

[69] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The 37th Asilomar SSC*, volume 2, pages 1398–1402. Ieee, 2003. 2.4.5, 3.4.4

[70] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004. 3.4.4

[71] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9421–9431, 2021. 2.2.1

[72] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, et al. Point-NeRF: Point-based Neural Radiance Fields. In *Proc. IEEE/CVF CVPR*, pages 5438–5448, 2022. 2.1, 2.2.1, 3.1

[73] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 2.1

[74] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, et al. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *Proc. IEEE/CVF ICCV*, pages 5752–5761, 2021. 2.2.1, 3.1

[75] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2.1, 2.2.1

[76] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. IEEE/CVF CVPR*, pages 586–595, 2018. 2.4.5, 3.4.4

[77] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 2.1