Towards Realism, Controllability, and Affordability in Learning-Based 3D Head Creation

by

Peizhi Yan

B.Sc., Algoma University, 2018

B.Eng., University of Jinan, 2019

M.Sc., Lakehead University, 2020

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Electrical and Computer Engineering)

The University of British Columbia

(Vancouver)

May 2026

© Peizhi Yan, 2026

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Towards Realism, Controllability, and Affordability in Learning-Based 3D Head Creation

submitted by **Peizhi Yan** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Electrical and Computer Engineering**.

Examining Committee:

Rabab K. Ward, Professor Emeritus, Electrical and Computer Engineering, UBC
Co-supervisor

Shan Du, Assistant Professor, Computer Science, Math, Physics & Statistics, UBC (Okanagan)
Co-supervisor

Z. Jane Wang, Professor, Electrical and Computer Engineering, UBC
Supervisory Committee Member

Dinesh Pai, Professor, Computer Science, UBC
University Examiner

Xiaoxiao Li, Associate Professor, Electrical and Computer Engineering, UBC
University Examiner

Farida Cheriet, Full Professor, Computer Engineering and Software Engineering, POLYTECHNIQUE MONTRÉAL
External Examiner

Abstract

Digital 3D head models are widely used in many applications, including video games, virtual reality, telepresence, and film production. There are three main objectives (also challenges) in 3D head creation: controllability, realism, and affordability. Although existing methods can achieve individual objectives, it remains challenging to satisfy multiple objectives together. For example, the 3D head models used in films can be photo-realistic, but they are very costly. In this thesis, we study how to achieve these objectives in 3D head creation through learning-based methods.

We start with the controllability objective and address 3D face shape creation that allows face-editing and stylization. We first propose the NEO-3DF approach, which uses sub-models to represent semantic facial parts (e.g., the nose). This design enables intuitive editing and improves alignment of the created 3D face with the 2D input image. Then, we introduce a weakly supervised 3D face stylization method that requires only one template face (artist-designed style) and transfers facial features from a reconstructed 3D face to this template. Besides controllability, these methods also enable affordability in 3D face reconstruction, editing, and stylization.

Next, we study how to achieve the realism objective and address 3D-consistent face image synthesis. To this end, we propose the StyleMorpheus method, which is trained with the objective of generating photorealistic face images. StyleMorpheus enables independent control over facial identity, expression, and appearance through a disentangled style-code space. The training of StyleMorpheus requires only in-the-wild 2D face images, which enables affordability by lowering the data collection costs.

Finally, we extend our study to jointly achieve controllability, realism, and affordability by addressing photorealistic and animatable 3D head avatar creation. We propose two 3D Gaussian Splatting (3DGS)–based methods: Déjà-vu and ArchitectHead. Déjà-vu first learns a generalized reconstruction model from 2D face images, which is then used to provide a strong initialization for efficient personalization using a short monocular video of a given individual. ArchitectHead further addresses the run-time trade-off between rendering quality and efficiency in 3DGS-based head avatars. Together, these methods enable the creation of highly realistic, controllable, and cost-effective 3D head avatars suitable for real-time applications.

Lay Summary

Digital 3D head creation is important for video games, virtual reality (VR), and film production. Modern techniques can produce realistic results. However, creating high-quality 3D heads that are easy to edit, manipulate, and at a low cost remains challenging. This thesis develops learning-based methods that create realistic-looking 3D heads that are easy to manipulate and at a reasonable cost.

First, we focus on the ability to manipulate a created 3D head in facial geometry modeling. We propose novel approaches to represent and edit face geometry and enable face stylization. Second, we show how real-world face images can be used to generate more realistic, editable, and 3D-consistent face images. Finally, we present methods for creating animatable 3D head avatars that capture realistic details and facial expressions, using only short video clips of real people.

Together, these contributions make high-quality digital head creation more practical and accessible.

Preface

This thesis is based on a collection of the candidate's manuscripts that have been published in journals, included in conference proceedings, or submitted for publication. To ensure consistency, these manuscripts have been modified as needed. The candidate was the primary author, with valuable input, suggestions, and revisions from co-authors of the manuscripts.

Chapter 3 is based on the following manuscripts:

- **P. Yan**, J. Gregson, Q. Tang, R. Ward, Z. Xu, and S. Du. NEO-3DF: Novel Editing-Oriented 3D Face Creation and Reconstruction. In *Proceedings of the Asian Conference on Computer Vision (ACCV). pages 486-502, 2022.*

  Peizhi Yan formulated the problem, designed and implemented the proposed method, and authored the paper. James Gregson contributed to the implementation of the differentiable ARAP algorithm. All the co-authors provided valuable feedback and suggestions throughout the algorithm development, experimentation, and manuscript writing process.

- **P. Yan**, R. Ward, Q. Tang, and S. Du. (2025). Neural 3D Face Shape Stylization Based on Single Style Template via Weakly Supervised Learning. *IEEE Transactions on Visualization and Computer Graphics (TVCG). VOL. 31, NO. 10, pages 9522-9529, October 2025.*

  Peizhi Yan formulated the problem, designed and implemented the proposed method, and authored the paper. All the co-authors provided valuable feedback and suggestions throughout the algorithm development, experimenta-

tion, and manuscript writing process.

Chapter 4 is based on the following manuscripts:

- **P. Yan**, R. Ward, D. Wang, Q. Tang, and S. Du. Learning Disentangled Features for NeRF-based Face Reconstruction. In *Proceedings of the 2023 IEEE International Conference on Image Processing (ICIP). 2023.*

  Peizhi Yan formulated the problem, designed and implemented the proposed method, and authored the paper. All the co-authors provided valuable feedback and suggestions throughout the algorithm development, experimentation, and manuscript writing process.

- **P. Yan**, R. Ward, D. Wang, Q. Tang, and S. Du. (2025). StyleMorpheus: Learning a StyleGAN-based 3D-aware morphable face model with a disentangled style space. *Elsevier Neurocomputing. Volume 654, 131329. 14 November 2025*.

  Peizhi Yan formulated the problem, designed and implemented the proposed method, and authored the paper. All the co-authors provided valuable feedback and suggestions throughout the algorithm development, experimentation, and manuscript writing process.

Chapter 5 is based on the following manuscripts:

- **P. Yan**, R. Ward, Q. Tang, and S. Du. Gaussian Déjà-vu: Creating Controllable 3D Gaussian Head-Avatars with Enhanced Generalization and Personalization Abilities. In *Proceedings of the 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2025*. (Accepted in round 1 for oral presentation, with an acceptance rate of 12%.)

  Peizhi Yan formulated the problem, designed and implemented the proposed method, and authored the paper. All the co-authors provided valuable feedback and suggestions throughout the algorithm development, experimentation, and manuscript writing process.

- **P. Yan**, R. Ward, Q. Tang, and S. Du. Estimating Virtual Camera FOV to Reduce Perspective Shape Distortion in 2D-to-3D Face Reconstruction. In *Proceedings of the 2025 IEEE International Conference on Image Processing (ICIP). 2025.*

  Peizhi Yan formulated the problem, designed and implemented the proposed method, and authored the paper. All the co-authors provided valuable feedback and suggestions throughout the algorithm development, experimentation, and manuscript writing process.

- **P. Yan**, R. Ward, Q. Tang, and S. Du. ArchitectHead: Continuous Level of Detail Control for 3D Gaussian Head Avatars. In *Proceedings of the 2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). 2026.*

  Peizhi Yan formulated the problem, designed and implemented the proposed method, and authored the paper. All the co-authors provided valuable feedback and suggestions throughout the algorithm development, experimentation, and manuscript writing process.

GENAI (specifically ChatGPT) was used in a limited capacity to assist with writing some of the computer code for this research and to proofread and improve the clarity of the thesis writing.

Table of Contents

Abstract . . . . . **iii**

Lay Summary . . . . . **v**

Preface . . . . . **vi**

Table of Contents . . . . . **ix**

List of Tables . . . . . **xiv**

List of Figures . . . . . **xvii**

List of Abbreviations . . . . . **xxii**

Acknowledgments . . . . . **xxiv**

Dedication . . . . . **xxv**

1 Introduction **1**

1.1 A Brief History of 3D Head Technologies 2

1.2 Motivation and Research Objectives 5

1.2.1 Challenges in 3D Head Creation 5

1.2.2 Research Objectives 8

1.3 Research Problems Formulation 9

1.4 Contributions 11

1.4.1 3D Mesh-Based Face Reconstruction, Editing, and Stylization . . . 11
1.4.2 3D-Aware Face Reconstruction and Synthesis . . . 13
1.4.3 Photorealistic and Animatable 3D Head Avatar Creation . 15
1.5 Overview of the Thesis . . . 16

2 Background . . . 19
2.1 Overview of Relevant 3D Representations and Their Rendering Techniques . . . 20
2.1.1 3D Mesh and Differentiable Rasterization . . . 21
2.1.2 Neural Radiance Field (NeRF) . . . 25
2.1.3 3D Gaussian Splatting (3DGS) . . . 26
2.2 3D Morphable Face/Head Models (3DMMs) . . . 28
2.3 Deep Generative Modeling . . . 29
2.3.1 Variational Autoencoder . . . 31
2.3.2 Autodecoder with Prior . . . 33
2.3.3 Generative Adversarial Network . . . 34
2.3.4 Diffusion Model . . . 35

3 Mesh-Based 3D Face Creation . . . 37
3.1 Introduction . . . 37
3.2 Related Works . . . 40
3.2.1 3D Face Modeling and Reconstruction . . . 40
3.2.2 3D Face Stylization . . . 41
3.3 Method 1: Editing-Oriented 3D Face Reconstruction (NEO-3DF) 42
3.3.1 Overview . . . 42
3.3.2 Background . . . 43
3.3.3 The Proposed Method . . . 44
3.3.4 Experiments . . . 54
3.3.5 Summary . . . 60
3.4 Method 2: Weakly Supervised Neural 3D Face Shape Stylization . 61
3.4.1 Overview . . . 61
3.4.2 Background . . . 62

3.4.3 The Proposed Method 64
3.4.4 Curvature Preservation 67
3.4.5 Experiments 70
3.4.6 Summary 78
3.5 Chapter Summary 79

4 3D-Aware Face Modeling 81
4.1 Introduction 81
4.2 Related Works 84
4.2.1 Parametric 3D Face Modeling 84
4.2.2 NeRF-Based 3D-Aware Morphable Face Models 84
4.2.3 StyleGAN-Based 3D-Aware Face Models 85
4.3 Method 1: Learning Disentangled Features for NeRF-based Face Reconstruction 87
4.3.1 Overview 87
4.3.2 The Proposed Method 87
4.3.3 Experiments 91
4.3.4 Summary 94
4.4 Method 2: A StyleGAN-Based 3D-Aware Morphable Face Model with a Disentangled Style Space (StyleMorpheus) 95
4.4.1 Overview 95
4.4.2 The Proposed Method 96
4.4.3 Experiments 103
4.4.4 Applications 115
4.4.5 Limitations and Future Work 120
4.4.6 Summary 120
4.5 Chapter Summary 121

5 Photorealistic 3D Head Avatar Creation 123
5.1 Introduction 124
5.2 Related Works 126
5.2.1 NeRF-Based 3D Face and Head Models 126
5.2.2 3D Gaussian-Based Head Avatars 127

5.3 Method 1: Creating Controllable 3D Gaussian Head-Avatars with Enhanced Generalization and Personalization Abilities (Déjà-vu) . 129
5.3.1 Overview . . . 129
5.3.2 The Proposed Method . . . 132
5.3.3 Experiments . . . 137
5.3.4 Limitations . . . 147
5.3.5 Summary . . . 147
5.4 Method 2: Continuous Level of Detail Control for 3D Gaussian Head Avatars (ArchitectHead) . . . 148
5.4.1 Overview . . . 148
5.4.2 Background . . . 150
5.4.3 The Proposed Method . . . 151
5.4.4 Experiments . . . 155
5.4.5 Limitations . . . 162
5.4.6 Summary . . . 163
5.5 Method 3: FOV Estimation for 3D Face Reconstruction — An Application of 3DGS-Based Head Models in Synthetic Data Generation 164
5.5.1 Overview . . . 164
5.5.2 Background . . . 165
5.5.3 The Proposed Method . . . 165
5.5.4 Experiments . . . 171
5.5.5 Summary . . . 175
5.6 Chapter Summary . . . 176

6 Conclusion and Discussion . . . 177
6.1 Contribution . . . 178
6.2 Limitations . . . 181
6.3 Ethics Discussions . . . 181
6.4 Future Work . . . 182
6.4.1 Balance Generalization and Personalization in 3D Head Avatar Creation . . . 182
6.4.2 Continuous Learning in 3D Head Avatar Personalization . 183

6.4.3 Using Multi-Modal Input Signals to Drive 3D Head Avatar Animation 184

Bibliography 185

Appendices 212

Appendix A: NEO-3DF 212

A.1 NEO-3DF Network Architectures 212

A.2 Estimate Right-Hand-Side in ARAP 214

A.3 List of Selected Facial Features for Editing 216

Appendix B: Déjà-vu 218

B.1 UV Gaussian Map in Déjà-vu Framework 218

B.2 Details of Datasets Used in Déjà-vu Framework 218

B.3 Comparison of Déjà-vu and HeadNeRF in Novel View Synthesis 221

Appendix C: ArchitectHead 222

C.1 ArchitectHead Framework Details 222

C.2 ArchitectHead Additional Results 225

List of Tables

Table 3.1 Comparison of the NEO-3DF and other parts-based 3D face methods. 45
Table 3.2 Mathematical notations for differentiable ARAP. 52
Table 3.3 3D-to-2D alignment results on validation data. 56
Table 3.4 Single-sample automatic 3D part shapes adjustment results. . . 60
Table 3.5 Mesh topology specifications. 71
Table 3.6 Stylization time comparisons. 72
Table 3.7 Ablation study on regularization loss terms. 76
Table 3.8 Reconstruction mean per-vertex Euclidean distance. 78

Table 4.1 Quantitative comparison of reconstruction. 92
Table 4.2 Per image fitting time comparison. 94
Table 4.3 Ablation study results on proposed part-based loss term. 94
Table 4.4 Comparisons in reconstruction performance (Facial Region Only). Evaluated on the CelebAMask-HQ dataset. 107
Table 4.5 Comparisons in reconstruction performance (Including Hair). Evaluated on the CelebAMask-HQ dataset. 107
Table 4.6 Comparisons in reconstruction performance (Facial Region Only). Evaluated on FFHQ dataset. 107
Table 4.7 Comparisons in reconstruction performance (Including Hair). Evaluated on FFHQ dataset. 107
Table 4.8 Code space normality evaluation results. 110
Table 4.9 Ablation study on the loss terms. 112
Table 4.10 Ablation study on the input style code. 114

Table 4.11 Frames-per-second of different 3D-aware face models on different devices. 119

Table 5.1 Comparison of supported features. 130
Table 5.2 Reconstruction performances (CelebAMask-HQ). 141
Table 5.3 Reconstruction performances (FFHQ). 141
Table 5.4 Average quantitative comparison across all three subjects of the PointAvatar dataset. 143
Table 5.5 Ablation Study on Training Data: Reconstruction performances (CelebAMask-HQ). 144
Table 5.6 Ablation Study on Training Data: Reconstruction performances (FFHQ). 144
Table 5.7 Quantitative comparisons of self-reenactment results on PointAvatar dataset. 158
Table 5.8 Quantitative comparisons of self-reenactment results on INSTA dataset. 158
Table 5.9 Rendering FPS (frames per second) performance comparison across LOD settings, GPUs, and floating-point precisions. . . . 161
Table 5.10 Quantitative comparison across different ablation settings and LODs. 162
Table 5.11 Mean point-to-surface distance between ground-truth point clouds and reconstructed meshes. 173
Table 5.12 Mean vertex distance. 174

Table A.1 Shape Encoder Network (E_i) Architecture 212
Table A.2 Latent Regressor Network (F_i) Architecture 213
Table A.3 Shape Decoder Network (D_i) Architecture 213
Table A.4 Offset Regressor Network Architecture 213
Table A.5 List of facial features selected for editing. 216
Table C.6 Summary of decoder networks. 224
Table C.7 Quantitative comparisons of self-reenactment results on the PointAvatar dataset. 225

Table C.8 Quantitative comparisons of self-reenactment results on INSTA dataset. 226
Table C.9 Evaluation of ArchitectHead on PointAvatar dataset. 228
Table C.10 Evaluation of ArchitectHead on INSTA dataset. 228

List of Figures

Figure 1.1 An example 3D head mesh. 1
Figure 1.2 Timeline of milestones in 3D head technologies. 4
Figure 1.3 The "uncanny valley." 6
Figure 1.4 Meta's codec avatar solution. 7
Figure 1.5 The relationships among three challenges in 3D head creation. 8
Figure 1.6 Simplified illustration of the 3D face/head reconstruction and rendering process. 11
Figure 1.7 Overview of the main contributions of this thesis. 18

Figure 2.1 Example types of 3D representations. 20
Figure 2.2 An example tri-mesh with two facets. 22
Figure 2.3 Example barycentric interpolation of vertices colors. 22
Figure 2.4 An example rendered 3D mesh and its UV texture map. 23
Figure 2.5 Workflow of a differentiable mesh renderer. 24
Figure 2.6 Comparison of traditional rasterization (non-differentiable) and differentiable soft rasterization. 24
Figure 2.7 A simplified diagram demonstrating the core mechanism of a Neural Radiance Field (NeRF). 25
Figure 2.8 An example illustrating the 3D Gaussian Splatting (3DGS) process. 27
Figure 2.9 Characteristics (sampling speed, generation quality, and mode coverage) of the three important generative modeling methods: GANs, VAEs, and diffusion models. 30
Figure 2.10 Illustration of an Autoencoder (AE). 31

Figure 2.11 Illustration of a Variational Autoencoder (VAE). 32
Figure 2.12 Illustration of an Autodecoder (AD). 33
Figure 2.13 Illustration of a Generative Adversarial Network (GAN). . . . 34
Figure 2.14 Illustration of a diffusion model (adapted from [70]). 35

Figure 3.1 Demonstration of our intuitive face editing results. 42
Figure 3.2 Flowchart of the proposed NEO-3DF. 45
Figure 3.3 Segments of our part-based face model. 46
Figure 3.4 Flowchart of our differentiable ARAP-based blending module. 54
Figure 3.5 3D-to-2D semantic face segments alignment results. 57
Figure 3.6 Vertex location change heatmaps. 58
Figure 3.7 Nose shape editing comparison. 58
Figure 3.8 Reconstruction and further editing examples. 59
Figure 3.9 Ablation study on the offset regressor and blending module of our framework. 59
Figure 3.10 Demonstration of the vertex offsets translation. 65
Figure 3.11 Our neural network-based offsets translation framework for 3D face stylization. 66
Figure 3.12 Example of the Voronoi area. 68
Figure 3.13 Comparisons of the deformed mesh smoothed by the traditional Laplacian smoothing filter and the proposed smoothing filter. 69
Figure 3.14 Loss term computation time comparisons. 72
Figure 3.15 Comparison of landmark-guided deformation transfer (LDT) [134], our naive offset transfer baseline, and our proposed neural transfer method. 73
Figure 3.16 The effects of the first six principal components (PCs) on the decoded shapes. 74
Figure 3.17 Stylization results of reconstructed faces. 75
Figure 3.18 Qualitative results of ablation study on regularization loss terms. 76
Figure 3.19 Examples of editing the local face features on the stylized faces. 77

Figure 3.20 Examples of applying various expression blendshapes to stylized neutral face meshes (first column) across different facial identities. 78

Figure 4.1 The proposed reconstruction learning framework. 89
Figure 4.2 The proposed SegLite network. 89
Figure 4.3 Example qualitative face reconstruction results. 93
Figure 4.4 The proposed model architecture follows an auto-encoder structure. 97
Figure 4.5 The proposed StyleGAN-based decoder architecture. 100
Figure 4.6 Single image face reconstruction results. 105
Figure 4.7 Novel view synthesis of reconstructed faces. 106
Figure 4.8 Disentanglement evaluation through code replacement. 109
Figure 4.9 Similarity of the replaced feature to the target image. 109
Figure 4.10 Visualization of the probability distributions of the first five dimensions for each code group (identity, expression, texture, and lighting). 110
Figure 4.11 Ablation study on the loss function. 112
Figure 4.12 Image reconstruction quality metrics (LPIPS↓, SSIM↑, PSNR↑) versus varying camera pose noise strengths. 113
Figure 4.13 Qualitative comparison of reconstructed face images. 113
Figure 4.14 Ablation study on the input codes. 114
Figure 4.15 Disentangled style mixing of latent space. 116
Figure 4.16 Face morphing results. 117
Figure 4.17 Part color editing examples. 117
Figure 4.18 Our virtual reality demonstration. 119
Figure 4.19 An example of the failure case. 120

Figure 5.1 Demonstration of the proposed Déjà-vu framework. 129
Figure 5.2 Overview of our proposed Déjà-vu framework for 3D Gaussian-based head avatar creation. 132
Figure 5.3 Our UV map structure and UV position regularization weights. 133
Figure 5.4 Example UV Gaussian Map. 133

Figure 5.5 Effects of modifying the scales of 3D Gaussian points. 134
Figure 5.6 Our single-image-based reconstruction network architecture. . 135
Figure 5.7 Qualitative comparison results on single-image-based reconstruction. 140
Figure 5.8 Self-reenactment results. 142
Figure 5.9 Qualitative comparison of the convergence speed of video-based optimization. 143
Figure 5.10 Ablation results on view consistency regularization. 145
Figure 5.11 Ablation results on position and scale regularizations. 145
Figure 5.12 Ablation results on different levels of rectification. 146
Figure 5.13 Ablation study on UV position regularization weights. 146
Figure 5.14 Convergence with different initialization schemes. 147
Figure 5.15 Demonstration of the proposed ArchitectHead method. 148
Figure 5.16 Pipeline of ArchitectHead. 152
Figure 5.17 Qualitative comparisons of self-reenactment results. 157
Figure 5.18 Rendered novel views. 158
Figure 5.19 Qualitative results of cross-identity reenactment. 159
Figure 5.20 Qualitative results of varying LODs. 160
Figure 5.21 Qualitative results of ablation study. 163
Figure 5.22 Example rendered face images. 167
Figure 5.23 Showcase of our modeled 3D bare head meshes based on multi-view images. 168
Figure 5.24 The image of subject 210, its head shape and the extracted face shape. 168
Figure 5.25 Example Images from the NeRSemble-FOV Dataset. 169
Figure 5.26 Our fitting-based 3D face reconstruction pipeline. 171
Figure 5.27 Confusion matrices of the testing FOV estimation results on NeRSemble-FOV dataset. 172
Figure 5.28 Qualitative reconstruction results. 174
Figure 5.29 Ablation study results of the reconstructions on rendered images of the same face with two FOVs. 175

Figure A.1 Heatmaps of editing controllers. 217

Figure B.2 Our method for automatically creating the mouth interior. . . 218
Figure B.3 An example identity generated from PanoHead, with 25 rendered views. 219
Figure B.4 Age, gender, and emotion distributions evaluated on the synthetic dataset. 220
Figure B.5 We realign the raw FFHQ images to cover the entire head. . . 220
Figure B.6 Qualitative comparison with HeadNeRF across varying viewing angles. 221
Figure C.7 Architecture of the mapping network $\mathcal{M}$. 223
Figure C.8 Architecture of the decoder network. 223
Figure C.9 Cross-identity reenactment results. 227
Figure C.10 Self-reenactment results with LOD 0 and LOD 1.0. 229

List of Abbreviations

2D	Two-dimensional
3D	Three-dimensional
3DMM	3D morphable model
3DGS	3D gaussian splatting
AD	Autodecoder
AE	Autoencoder
ARAP	As-rigid-as-possible mesh optimization
BFM	Basel Face Model
CBD	Cage-based deformation
CG	Computer graphics
CNN	Convolutional neural network
CV	Computer vision
DDPM	Denoising Diffusion Probabilistic Models
DOF	Degrees of freedom
ELBO	Evidence Lower Bound
FOV	Field-of-view
FPS	Frames per second
GAN	Generative adversarial network
GENAI	Generative AI

GPU	Graphic processing unit
KL	Kullback-Leibler
LBS	Linear blend skinning
LOD	Level of detail
MLP	Multi-layer perceptron
NERF	Neural radiance fields
PCA	Principal component analysis
SDF	Signed distance function
SIFR	Single image-based 3D face reconstruction
VAE	Variational auto-encoder
VR	Virtual reality
VRAM	Video random access memory

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisors, Prof. Rabab Kreidieh Ward and Prof. Shan Du, for providing exceptional guidance and support not only in my research but also in my academic life. I am also grateful to Prof. Jane Wang for her valuable advice and support as part of my supervisory committee. I am equally indebted to Dr. Qiang Tang, who provided invaluable guidance throughout my research.

I am deeply appreciative of my lab mates for their support. The discussions we had broadened my perspectives, and our friendship made my Ph.D. journey more enjoyable and meaningful. Special thanks go to Dr. Dan Wang for her insightful suggestions on my research projects. May our friendship continue to grow in the years ahead.

My sincere thanks go to all my teachers, friends, and everyone who has supported me along the way.

Last but not least, I express my deepest gratitude to my family for their unconditional love and unwavering support.

Dedication

To my family, teachers, and friends.

Chapter 1

Introduction

"The world around you might not be real.[1]*"*

– Christopher Nolan

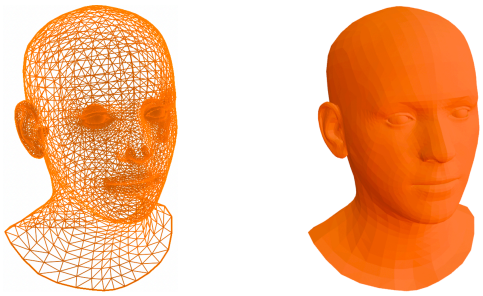


Figure 1.1: An example 3D head mesh generated from the FLAME head model [109]. Left: wireframe visualization showing the triangular mesh structure. Right: rendered surface geometry of the mesh.

For decades, academia and industries have pursued the goal of creating digital 3D head models that are both realistic in appearance and support animation. In the movies industry, these models are widely used to enable post-production effects,

[1] "The whole concept of avatars and living life as someone else, there's a relationship to what we're doing, but I think when I first started trying to make this film happen it was very much pulled from that era of movies where you had 'The Matrix,' you had 'Dark City,' you had 'The Thirteenth Floor' and, to a certain extent, you had 'Memento' too. They were based on the principles that the world around you might not be real."

such as making an actor's face appear younger or older than their actual age [203]. Furthermore, a digital replica of an actor's head enables head re-targeting, allowing the original actor's facial appearance to be transferred onto a substitute performer when the actor is unavailable [188]. Beyond filmmaking, digital 3D heads are fundamental in video games and virtual reality (VR), where they serve as the basis for digital humans and interactive characters (also known as virtual avatars). Such applications include virtual and immersive meetings [189], telepresence systems [31, 199], and virtual worlds such as the metaverse [190].

1.1 A Brief History of 3D Head Technologies

The first computer-generated 3D face model was introduced and developed by Frederick Parke in his Ph.D. research over fifty years ago (1972) [137, 138]. This model used a 3D mesh-based representation, where vertices and edges establish the topology of the facial surface. In 1985, the short film Tony de Peltrie featured the first computer-animated 3D human face in filmmaking [11]. Since then, the use of 3D faces and heads has expanded rapidly in both films and video games. Making the facial animation appear natural and realistic is challenging, as subtle changes in facial expression can be recognized by humans [137]. Terzopoulos and Waters developed a physically based approximation of facial tissue and muscles, and used video sequences of real human faces to drive the animation of a 3D face [183]. In 2009, the movie "Avatar" showcased a breakthrough in facial performance capture technology, which enabled the creation of highly expressive and lifelike humanoid characters ("Avatars") [60]. Its release marked a significant advancement in 3D head modeling and facial animation.

While industries largely depend on 3D artists to meticulously sculpt/model 3D heads, researchers have consistently explored more automated approaches using scans of real individuals. In 1998, Blanz and Vetter introduced a method that synthesized novel 3D faces from samples of individual 3D face scans [191]. The key idea was to first register all scanned faces to a reference 3D face topology, producing a dataset of examples of 3D faces with a uniform mesh structure (topology). Then, new faces can be generated by linearly blending these samples in the dataset. In the following year (1999), they improved the method and proposed the 3D Mor-

phable Model (3DMM) [14]. The 3DMM utilized principal component analysis (PCA) to extract statistically significant modes of variation (principal components) from their 3D face dataset. Therefore, new faces can be synthesized from these components, given the corresponding coefficients (similar to blending weights). The 3DMM serves as a robust prior for 3D face modeling, enabling the creation of realistic faces and providing a foundation for decades of subsequent research [45].

An important application of the 3DMM is in image- or video-based 3D face and head reconstruction. Depending on the type of input data, these reconstruction tasks can be categorized into four groups: (1) single-image-based reconstruction [14, 38, 48, 248], (2) monocular video-based reconstruction [6, 56], (3) multi-view image-based reconstruction [207], and (4) multi-view video-based reconstruction [211]. The first category relies on very limited information (a single image), while the latter ones exploit richer information about the 3D head. Multi-view images or videos provide implicit constraints on the 3D structure, and video sequences are particularly valuable for capturing temporal dynamics such as facial expressions. However, collecting multi-view data typically requires complex setups such as camera rigs [93] or light stages [122], which are expensive and impractical for large-scale use. In this thesis, we primarily focus on single-image and monocular video-based approaches. These approaches enable data collection using consumer-level devices, such as smartphones, thereby greatly reducing the cost of 3D head reconstruction.

Over the past decade, deep learning–based methods have been applied in 3D face and head reconstruction, enabling the recovery of 3D face geometry and texture directly from images or videos [38, 48, 235, 248, 250]. These methods leverage large-scale training data to train deep neural networks to recognize complex patterns in the relationship between 2D and 3D faces. The benefit is that, after training, the reconstruction process is very fast because it typically does not require iterative-based optimization, such as analysis-by-synthesis [14]. A common component in these methods is an image encoder, usually a convolutional neural network (CNN). The image encoder learns to map 2D images to a latent space that captures the essential information to represent the face. These latent features are then used in the subsequent reconstruction process. The most used approach is model-based (i.e., 3DMM-based), which trains the network model to estimate

the 3DMM coefficients [38, 235]. Other approaches employ a decoder network to directly regress the 3D face shape [47, 186, 216]. To improve photorealistic quality, some methods use generative adversarial learning to synthesize high-quality texture maps (e.g., advanced texture maps used for physically-based rendering) [57, 192].

In recent years, there has been a growing trend toward exploring the emerging 3D representations as alternatives to traditional 3D meshes in head modeling. Approaches include neural signed distance functions (neural SDF) [237], neural radiance fields (NERF) [8, 71, 173], and 3D Gaussian splatting (3DGS) [145, 210, 219]. These representations overcome the rigidity of the mesh-based models. They enable more flexible descriptions of complex features, such as facial hair. Because the models of these representations are inherently differentiable, 3D head models can be reconstructed directly from 2D images or videos, using gradient-based optimization. This ability to reconstruct photorealistic heads directly from visual input makes these representations particularly suitable for creating digital 3D head avatars. Because the optimization objective is based on rendered image fidelity, the resulting reconstructions are typically photorealistic. As a result, 3D head avatars have become a popular research topic and are drawing increasing interest in both academia and industry [110, 122, 144, 153, 219].

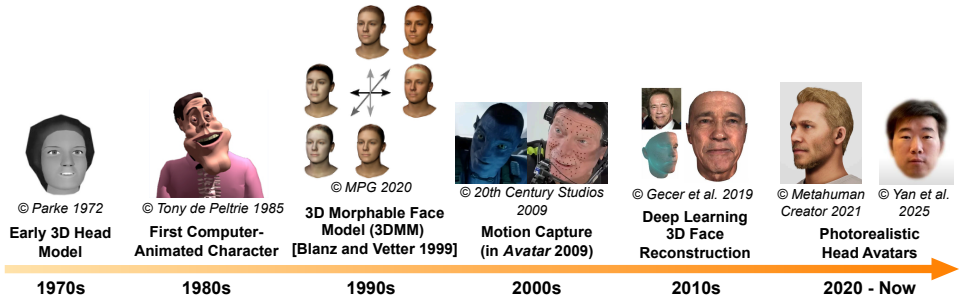


Figure 1.2: Timeline of milestones in 3D head technologies. From left to right, the example images are adapted from [137], [11], [109], [60], [57], [187], and [219].

With continued advancements in graphics hardware (e.g., GPU), software, and modeling techniques, the creation and use of photorealistic 3D digital heads are

now feasible in real-time applications such as video games, paving the way for the next generation of realistic 3D head avatars. In Figure 1.2, I summarize some milestones in 3D head technologies.

1.2 Motivation and Research Objectives

1.2.1 Challenges in 3D Head Creation

Although industries already employ advanced hardware, algorithms, and software to create digital 3D head models, significant challenges remain to be addressed. In the following, I summarize the most important challenges in three aspects: (1) **realism**, (2) **controllability**, and (3) **affordability**.

Realism concerns whether a 3D head appears convincingly realistic. For a generic 3D digital human head, it must resemble a real human. For a digital replica of a specific person, it must not only resemble a human but also accurately reconstruct the individual's identity. Realism can be further divided into *static* and *dynamic* aspects. Static realism is relatively easier to achieve, as it only requires the head to look realistic in a still form (e.g., a rendered 2D still image of a 3D head), without considering motion. Dynamic realism, on the other hand, concerns whether the movements are natural and consistent with the person being represented (e.g., facial expressions and speech-related motions). Because facial features are crucial for recognizing individuals, the human visual system is highly sensitive to subtle differences and unnatural traces on a face. In 1970, Mori first described the phenomenon that people's reactions to humanoid robots do not change monotonically with increasing human likeness [131]. He introduced the term "uncanny valley" to describe the sudden drop in familiarity (or affinity) that occurs when a robot appears nearly human but still exhibits subtle details revealing that it is not (see Figure 1.3). Therefore, achieving realism in a digital 3D head is highly challenging, as minor imperfections can cause discomfort.

Controllability is another major challenge in digital 3D head creation. Controllability here lies in the ability to modify (edit) the identity and facial features (after constructing the head). Users often wish to customize the reconstructed 3D head [172, 216]. For example, by editing (modifying) facial features to make the

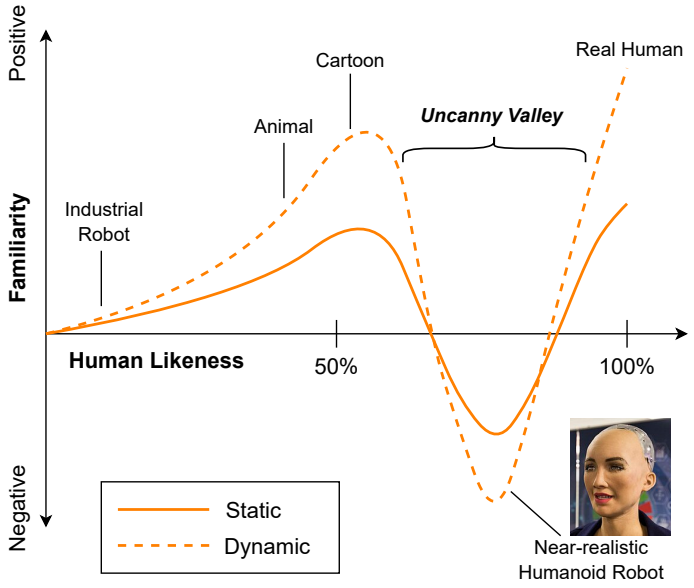


Figure 1.3: The relationship between human familiarity (or affinity) and the perceived human likeness of an object (adapted from [131, 138]). The sudden drop in familiarity as the object approaches human realism is known as the "uncanny valley." "Static" and "dynamic" represent objects presented without motion (e.g., a still statue of a head) and with motion cues (e.g., an animated head with facial expressions), respectively. The image of the humanoid robot "Sophia" was cropped from the original image by ITU Pictures, Geneva, Switzerland (https://www.flickr.com/photos/itupictures/27254369347/).

reconstructed head more plausible or to convey a different types of emotions. Controllability requires prior knowledge of human head structure to ensure that the edited features remain coherent and do not result in unnatural appearances. Another aspect of controllability is the animation of the 3D head. Achieving realistic animation requires modeling facial and head dynamics, such as facial expressions. One approach is through physical simulation, which relies on accurate modeling of the skin, underlying muscles, and skull [204]. Another approach is data-driven, where facial movements of real humans are captured and machine learning approaches are used to recognize patterns and synthesize varying and realistic facial

dynamics [222]. The physical model of a human head is very complex. Therefore, achieving realistic control over 3D head models is a challenging task.

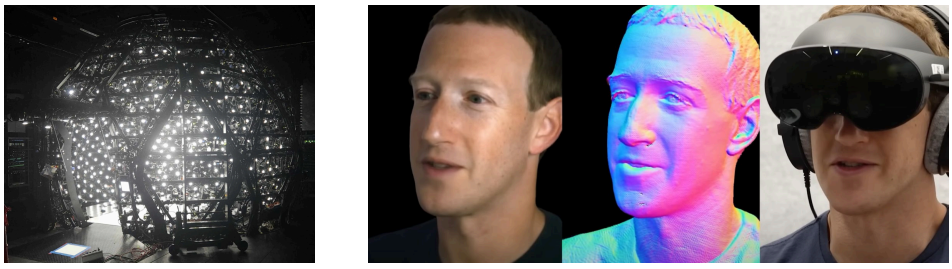


Figure 1.4: Meta's codec avatar solution. Left: the light stage setup at Meta's codec avatars lab (photo from [199]). Right: Mark Zuckerberg (2023) using a VR device in a telepresence application (image adapted from Lex Fridman's video, https://www.youtube.com/watch?v=MVYrJJNdrEg).

The last challenge is the **Affordability** of a digital 3D head. In this thesis, affordability refers to the overall cost associated with both the creation of a 3D head model and its practical use in downstream applications. In film production, budgets are usually much higher than in other domains. For example, in the movie "Fast & Furious 7," the estimated cost of creating a 3D head for the late actor Paul Walker and producing the final rendered video frames was around $50 million [44]. In other applications, such as video games, where real-time efficiency is critical, the cost of rendering a 3D head into video frames becomes a major concern. Unlike film production, which can rely on large-scale rendering farms and offline processing, consumer devices are constrained by limited computational resources. Furthermore, while film studios only need to create 3D head models for a few specific identities, it is not cost-effective to generate high-quality 3D head models for a large user base. In recent years, more automatic approaches have been developed to capture a person's data and reconstruct their avatar model. For example, Figure 1.4 shows the data collection setup and the avatar of Mark Zuckerberg created by Meta's codec avatar lab. Although methods such as light stages or multi-view camera rigs enable the capture of abundant data to reconstruct photorealistic avatars, the required setup and processing remain too costly for large-scale

deployment. As a result, balancing quality and affordability remains a critical challenge.

Figure 1.5 summarizes the relationships among the three major challenges discussed above. These challenges inherently conflict with one another. For example, in the movie industry, it is preferred to achieve extreme realism at the expense of affordability. Existing methods generally optimize for only one or two of these aspects, making them unsuitable for applications where all three are essential. This limitation becomes particularly evident in scenarios such as virtual reality, where a large number of users each require their own high-quality and controllable 3D digital avatar. Therefore, there is a strong need for new approaches that jointly consider all three challenges, with the ultimate goal of making 3D head creation more practical and accessible.

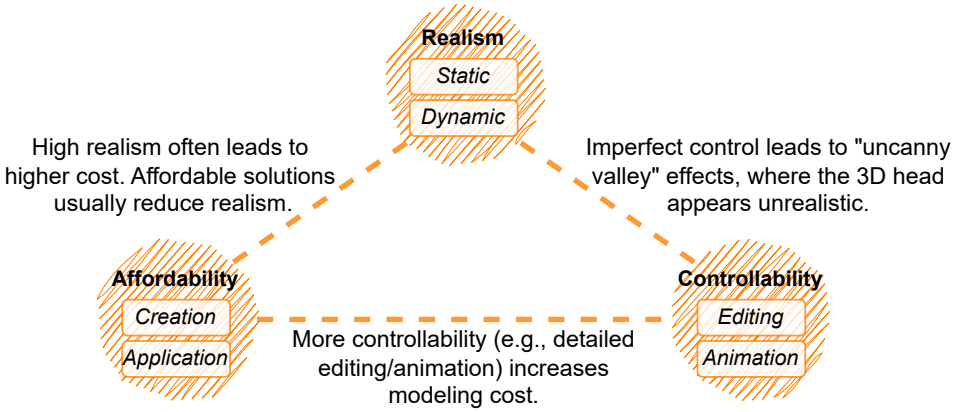


Figure 1.5: The relationships among three challenges in 3D head creation.

1.2.2 Research Objectives

The goal of this thesis is to employ data-driven and learning-based approaches to develop methods for 3D digital human head construction. The construction must satisfy three major concerns. The constructed head must have a realistic appearance (realism); the methods must allow controllability over the constructed head's features and motion (controllability); and the construction and application costs must be reasonably low (affordability).

Machine learning plays a central role in this work, as it enables us to model complex patterns in human head geometry, appearance, and motion. We also specifically focus on reconstructing a 3D head whose features can be explicitly controlled, and the reconstruction employs only a single 2D image or a monocular video. This is because 2D images and monocular videos are easy to acquire and align with our goal of reducing costs. Our methods can benefit different user groups. For 3D artists, our work provides tools for the rapid prototyping of 3D heads. For general users with no prior background in 3D modeling, our approach enables them to create personalized 3D head avatars from a short video or a single image.

1.3 Research Problems Formulation

To achieve the objectives of realism, low cost, and controllability discussed in the previous section, it is necessary to examine the problem of 3D head creation from a fundamental modeling perspective. In this thesis, we argue that these objectives can be formulated as two coupled problems: generalization and personalization. In our context, generalization refers to whether a general 3D face or head model can be reused to represent a wide range of identities, which is essential for achieving low-cost scalability and controllability. Personalization, on the other hand, focuses on adapting such a general model to a specific individual in order to faithfully capture the individual's identity-specific characteristics and achieve high realism.

An effective approach to generalize from limited training data to unseen 3D faces or heads is to use a generative model. A generative model learns the underlying probability distribution of the data and can therefore synthesize new samples that resemble the training examples. Common approaches for learning such generative models include Variational Autoencoders (VAEs) [92], Generative Adversarial Networks (GANs) [63], and diffusion models [70].

The goal of training a generative 3D face or head model is to estimate the model parameters θ (which are different from the input parameters) so that the model captures the distribution of valid 3D faces or heads. We represent the generative model as a function

$$F_\theta : \alpha \rightarrow \hat{x}, \tag{1.1}$$

where α denotes the set of input parameters sampled from a prior distribution $p(\alpha)$, and $\hat{x}$ denotes the synthesized 3D face or head. The training objective is to learn θ such that the generated samples $\hat{x}$ follow the true data distribution $p_{\text{data}}(x)$, where x denotes the real 3D heads. In deep generative models such as VAEs, GANs, and diffusion models, the prior $p(\alpha)$ is typically fixed, most commonly as a standard normal distribution, while the generator (or the decoder in VAEs) is trained to map samples from this prior to the data distribution.

Although a generalized 3D face/head model can produce realistic results, it often struggles to capture the fine details and unique characteristics of a specific individual. Therefore, personalization is necessary when reconstructing a realistic 3D face or head for a particular person.

Given a generative 3D face/head model, the reconstruction of a new 3D face or head from given data I of a person's face/head can be viewed as inferring the model's input parameters α with respect to I. This process can be seen as a form of personalization, as it adapts a general 3D face or head model to a specific individual based on the observed data. In this thesis, I can be a single image or a video (sequence of images) of the face/head. According to Bayes' theorem, inference with the generative model corresponds to estimating the posterior distribution of the model's input parameters:

$$\underbrace{p(\alpha \mid I)}_{\text{posterior}} \propto \underbrace{p(I \mid \alpha)}_{\text{likelihood}} \underbrace{p(\alpha)}_{\text{prior}}. \tag{1.2}$$

In this formulation, the likelihood quantifies the fidelity of the reconstruction. It measures how well the reconstructed 3D face/head $\hat{x}$ matches the observed data I. The prior encodes our a priori belief about plausible parameter values and constrains the reconstructed 3D face or head to remain within a reasonable manifold of valid 3D faces or heads. Note that $\hat{x}$ is represented in 3D while I is observed in 2D. To enable end-to-end training without 3D supervision, a renderer is required to project $\hat{x}$ onto the 2D image plane, producing the rendered image(s) $\hat{I}$. In a gradient-based optimization framework, this renderer must be differentiable so that the reconstruction loss can be backpropagated to the parameters α during optimization. Figure 1.6 illustrates the simplified relationships among α, the generative

model, the reconstructed 3D face/head $\hat{x}$, and the rendered image(s) $\hat{I}$.

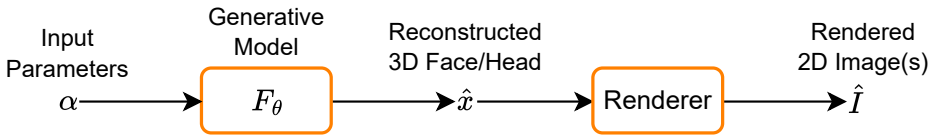


Figure 1.6: Simplified illustration of the 3D face/head reconstruction and rendering process.

Besides optimizing α based on I to achieve personalization, one can also further optimize the reconstructed 3D face or head $\hat{x}$ to better match the observation(s) I. However, such optimization is prone to overfitting and therefore requires carefully designed constraints and optimization algorithms.

1.4 Contributions

This thesis makes several novel contributions toward controllable, realistic, and affordable 3D head modeling. The proposed contributions address a series of closely related problems in 3D head creation in a progressive manner. We begin by studying controllable 3D face shape modeling based on mesh representations and introduce methods that support intuitive editing and stylization (**Chapter 3**). Building on this foundation, we then investigate 3D-aware face image synthesis, where models learn 3D-consistent rendering from 2D images to achieve improved realism and generalization (**Chapter 4**). Finally, we develop personalized and animatable 3D head avatar creation methods that jointly address realism, controllability, and affordability by leveraging the recently proposed 3D Gaussian Splatting representation (**Chapter 5**). In the following, we summarize the individual contributions of this thesis in detail.

1.4.1 3D Mesh-Based Face Reconstruction, Editing, and Stylization

We start with the 3D face because it is a crucial component of the human head, containing distinctive features that enable recognition of a person's identity. The 3D mesh is the most widely adopted representation in 3D face modeling and has served as a foundation for decades of research. While mesh-based 3D face recon-

struction has been extensively studied, comparatively less attention has been given to the editing and stylization of reconstructed 3D faces. In **Chapter 3**, we focus on improving controllability in mesh-based 3D face shape creation, with particular emphasis on editing and stylizing reconstructed 3D faces.

We propose the `NEO-3DF` framework for single-image 3D face reconstruction (3D face reconstruction from a single 2D image) with its subsequent editing ability (**Chapter 3.3**). Before this work, existing methods could reconstruct a 3D face from a single image, but they could not support flexible and local face editing because they relied on a single global face model. With NEO-3DF, we enable independent and intuitive editing of local facial parts, such as the eyes, nose, and lips, while maintaining geometric consistency. This is achieved through a face representation that is based on the different parts of a face, which is shown to improve model flexibility and allows local facial parts to be controlled independently. To ensure geometric consistency after editing, we present a differentiable deformation optimization mechanism based on the traditional As-Rigid-As-Possible (ARAP) method [170], which seamlessly blends a face parts after modifying it back into the full face mesh while enabling end-to-end training. As a result, NEO-3DF provides significantly greater editing flexibility and controllability than conventional global-based 3D face models, and achieves improved 3D-to-2D alignment in single-image 3D face reconstruction compared to existing methods. In summary, the main contributions of the proposed NEO-3DF are:

1. NEO-3DF is the first unified framework that bridges single-image 3D face reconstruction and intuitive 3D face editing;

2. We propose a semantic face parts-based 3D face model that enables independent and flexible editing of local face part;

3. We propose a differentiable version of the traditional ARAP method that ensures consistency in the face surface after being edited, and supports end-to-end optimization.

In **Chapter 3.4**, we propose `Neural 3D Face Stylization`, a weakly supervised approach for stylizing reconstructed 3D faces while preserving facial identity. Prior learning-based 3D face stylization methods typically rely on large

datasets of paired 3D faces and corresponding stylized 3D faces, which require expensive manual effort to construct. With our method, 3D face stylization can be learned using only a single artist-designed 3D face template to provide style information, significantly reducing the cost of data preparation. We formulate 3D face stylization as a deformation transfer problem [171], which preserves prominent facial geometry (e.g., face shape and nose structure) while adapting it to a stylized target mesh. To avoid expensive per-input optimization required by traditional deformation transfer methods [134, 171], we introduce a learning-based formulation with a novel template-guided mesh smoothing regularization loss, enabling efficient inference. Once trained, the proposed method achieves results comparable to traditional deformation transfer while being approximately $3,000\times$ faster. The proposed Neural 3D Face Stylization method can be seamlessly integrated with our NEO-3DF framework, enabling a unified pipeline for 3D face reconstruction, editing, and stylization. In summary, the main contributions of the proposed Neural 3D Face Stylization method are:

1. It is the first 3D face stylization method that requires only one artist-designed 3D stylized face template;

2. We propose a weakly supervised learning approach that eliminates the need for paired training data while preserving identity-related face geometry;

3. We propose a novel mesh regularization loss term to reduce the noise in the resulting stylized 3D face and preserve the structural characteristics of the artist-designed template mesh.

1.4.2 3D-Aware Face Reconstruction and Synthesis

While the contributions in Chapter 3 focus on controllability in mesh-based 3D face shape creation, such representations lack important components such as facial hair, eyeballs, teeth, and detailed appearance. To achieve improved realism, in **Chapter 4**, we propose 3D-aware face creation methods. In this thesis, 3D-aware refers to methods that learn 3D-consistent rendering from 2D images, with controllable camera poses. 3D-aware methods implicitly represent 3D information using a neural network model without explicitly modeling 3D geometry or texture

as in 3D meshes, which improves the flexibility in representing complex and rich details.

In **Chapter 4.3**, we propose an end-to-end learning-based framework for single-image face reconstruction using 3D-aware face models, demonstrated on HeadNeRF [71]. Before this work, HeadNeRF could not generalize well to single-image reconstruction, because it required direct optimization of latent codes, which often led to overfitting. We address this problem by introducing a dedicated face encoder that predicts the latent parameters of HeadNeRF from a single in-the-wild image, enabling improved generalization to unseen identities. In addition, we introduce lightweight semantic supervision, including a semantic face segmentation module and a facial-part color regularization loss, to further improve reconstruction quality. As a result, our framework enables effective single-image reconstruction with improved generalization and provides a practical way to evaluate and improve existing 3D-aware head models through end-to-end learning. However, while this framework improves generalization for single-image-based reconstruction, it still relies on existing 3D-aware face models (e.g., HeadNeRF), whose controllable modeling is based on expensive, lab-collected datasets and is inherently limited in diversity.

In **Chapter 4.4**, we propose `StyleMorpheus`, a 3D-aware morphable face model that generalizes well to unseen identities using only in-the-wild face images to train. Before this work, existing 3D-aware face models required expensive lab-collected datasets with multi-view and multi-expression images to achieve controllable and disentangled representations [71, 247]. StyleMorpheus is based on StyleGAN [88] and introduces a disentangled latent space that enables independent control of facial identity, expression, appearance, and lighting. As a result, we enable controllable 3D-aware face modeling without relying on costly lab-collected datasets, while achieving improved generalization to unseen identities. In summary, the main contributions of the proposed StyleMorpheus model are:

1. Unlike previous 3D-aware morphable face model approaches, StyleMorpheus can be trained using only in-the-wild 2D face images;

2. Unlike existing StyleGAN-based face models, StyleMorpheus enables independent control over facial identity, expression, and appearance through a disentangled style code space;

3. StyleMorpheus achieves superior photorealistic face rendering quality, surpassing existing 3D-aware morphable face models.

1.4.3 Photorealistic and Animatable 3D Head Avatar Creation

While the 3D-aware face models introduced in the previous chapter improve photorealism and generalization, they remain limited in terms of real-time rendering efficiency and personalization for animatable head avatars. In **Chapter 5**, we propose two new methods for photorealistic and animatable 3D head avatar creation, namely `Déjà-vu` and `ArchitectHead`. We achieve this by adopting 3D Gaussian Splatting (3DGS) [90], which supports real-time rendering while preserving high visual quality. Building on 3DGS and our earlier works on mesh-based and 3D-aware head modeling, Déjà-vu addresses fast personalization and realistic facial animation, while ArchitectHead enables continuous control over the level of detail (LOD) for efficient and scalable rendering.

In **Chapter 5.3**, we propose `Déjà-vu`, a 3DGS-based head avatar method that enables fast personalization and realistic facial animation. Existing 3DGS-based head avatar methods relied on naive initialization and required long training times for personalization [145, 210, 215]. Déjà-vu leverages a dedicated and generalized prior model to initialize 3D Gaussian points, enabling fast and stable personalization from monocular video. Specifically, we train a reconstruction model that predicts a 3DGS-based head from a single image, which is then refined using a short video sequence. To support realistic facial animation, we further incorporate expression-aware blendshapes inspired by traditional mesh-based animation. As a result, Déjà-vu achieves state-of-the-art visual quality while reducing personalization time to only a few minutes. In summary, the main contributions of Déjà-vu are:

1. Déjà-vu is the first method that supports reconstructing a 3DGS-based head from a single input image;

2. The single-image reconstruction model of Déjà-vu provides a better initialization for personalization using a short monocular video;

3. The proposed expression-aware blendshapes and its training approach enable

faster personalization training and improved facial animation quality.

In many applications, it is important to control the level of detail (LOD) to balance rendering efficiency and visual quality. However, effective LOD control is challenging in 3DGS-based head avatars due to the unstructured nature of Gaussian points. To address this, we propose `ArchitectHead`, the first 3DGS-based head avatar creation method that supports continuous LOD control (**Chapter 5.4**). The key idea is to parameterize 3D Gaussians in a 2D UV space, which enables resolution-aware resampling to control the number of Gaussian points. We further introduce a neural decoding scheme that predicts Gaussian attributes from latent features conditioned on the desired LOD. This design allows ArchitectHead to represent complex geometry and appearance consistently across different levels of detail. In summary, the main contributions of ArchitectHead are:

1. ArchitectHead is the first 3DGS-based head avatar method that supports continuous control of the level of detail (LOD) without retraining;

2. We propose learnable multi-level UV feature maps and an adaptive sampling strategy to efficiently control the LOD through the sampled UV map resolution.

1.5 Overview of the Thesis

In the present chapter (**Chapter 1**), I introduce the motivation and context for digital 3D head modeling, review its historical development, and discuss the current challenges (i.e., realism, controllability, and affordability). I also formulate the key research problems of generalization and personalization in 3D head modeling, and briefly summarize our contributions. The rest of this thesis is organized as follows.

In **Chapter 2**, I provide background knowledge on related 3D representations, existing 3D mesh-based face/head modeling approaches, and related generative modeling methods.

The main body of this thesis consists of three chapters: **Chapters 3** to **5**. In **Chapter 3**, we study 3D mesh-based face modeling with the aim of improving controllability in face shape manipulation. We propose a part-based 3D face model,

along with its applications in 3D face reconstruction and editing. We also introduce a neural network–based deformation transfer method for learning 3D face stylization.

In **Chapter 4**, we focus on improving realism by exploring NeRF-based 3D-aware modeling to address the limited representation capacity of 3D meshes. We propose a novel 3D-aware morphable face model that achieves realistic face image synthesis with strong generalization to unseen identities, controllability in editing, and affordability in its training data requirements.

In **Chapter 5**, we further improve the realism of representing individual-specific characteristics (personalization) while reducing rendering costs. We leverage the emerging 3D Gaussian Splatting (3DGS) technique to develop 3DGS-based head avatar creation methods that achieve photorealistic quality, controllability in animation, and affordability in rendering efficiency.

Finally, in **Chapter 6**, I summarize the contributions and limitations of the proposed methods, highlight possible directions for future work, and discuss potential ethical issues related to 3D head creation research.

Figure 1.7 provides an overview of the main contributions of this thesis.

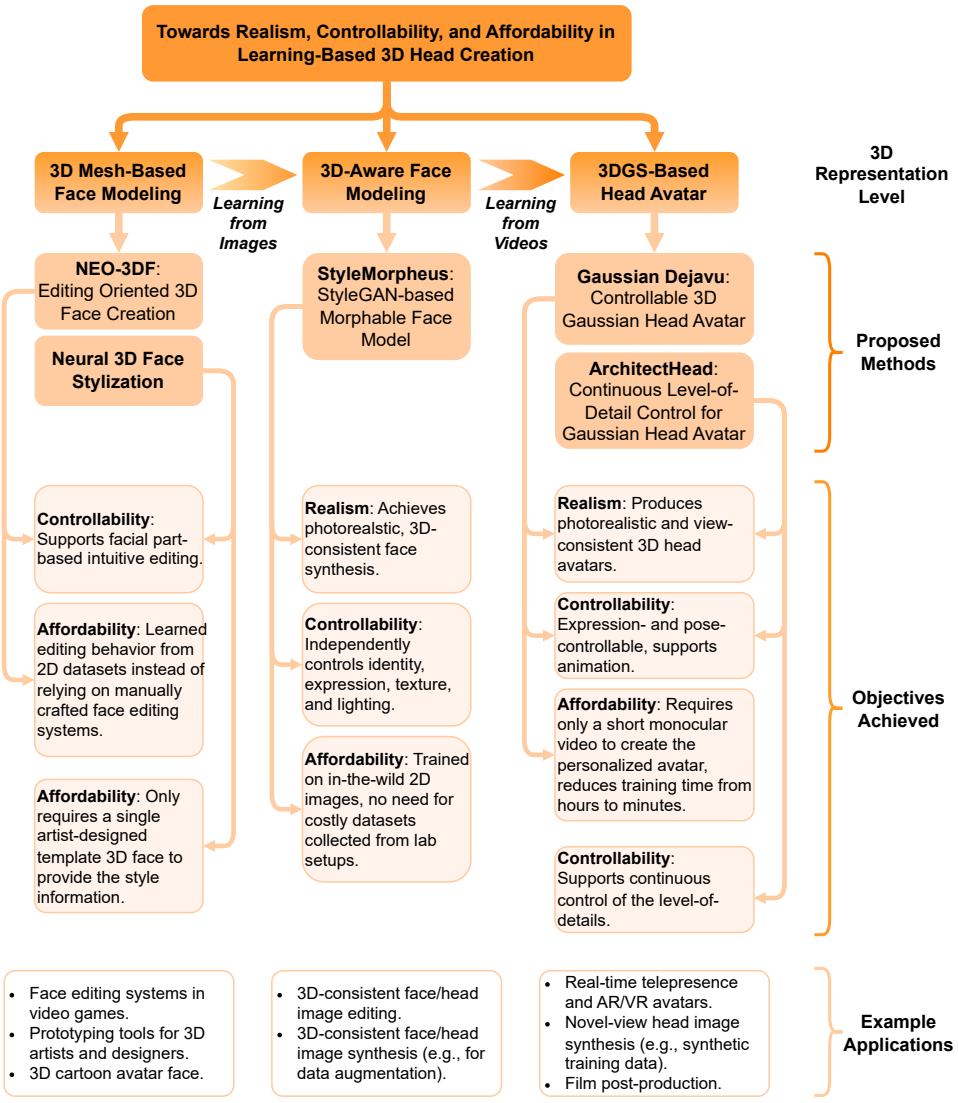


Figure 1.7: Overview of the main contributions of this thesis. The proposed methods are organized according to different 3D representations. Their key achievements in realism, controllability, and affordability are briefly summarized.

Chapter 2

Background

Our research methodology draws on multiple closely related fields, including computer vision (CV), 3D computer graphics (3D CG), and machine learning. These fields provide complementary perspectives on how visual information is represented, generated, and interpreted, and together form the technical foundation of this thesis.

Computer vision focuses on analyzing and interpreting the visual world from images and videos [177], whereas 3D computer graphics is concerned with the synthesis, modification, and visualization of virtual scenes and objects [17, 36]. The most common form of visualization in 3D computer graphics is 3D rendering, where a 2D image is generated from a 3D scene, similar to taking a photograph of a real scene [53]. From this perspective, 3D computer graphics can be viewed as a *forward process* that produces visual observations from known 3D scenes, while computer vision can be regarded as an *inverse process* that seeks to recover scene attributes (e.g., shape, lighting, appearance, and semantics) from visual data.

This chapter provides the necessary background on the important concepts and techniques used throughout this thesis. We first review common 3D representations and their associated rendering methods, with a focus on representations that support differentiable rendering (**Chapter 2.1**). We then introduce important 3D face and head modeling approaches (**Chapter 2.2**), followed by an overview of deep generative modeling methods that form the foundation of the proposed techniques (**Chapter 2.3**).

2.1 Overview of Relevant 3D Representations and Their Rendering Techniques

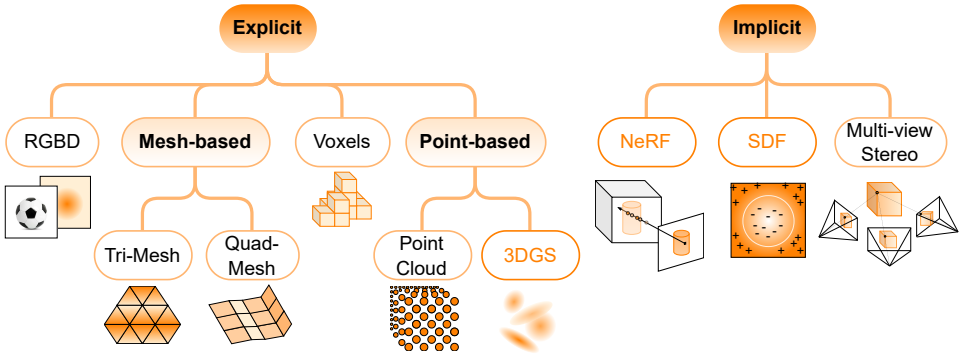


Figure 2.1: Example types of 3D representations grouped into explicit and implicit categories. It is worth noting that Signed Distance Function (SDF), Neural Radiance Field (NeRF), and 3D Gaussian Splatting (3DGS), are commonly realized through learning-based approaches.

There is a wide variety of 3D representations in computer graphics, as no single representation can satisfy the diverse requirements of different applications [1, 200]. Based on whether the 3D geometries (and textures/colors) are directly stored, 3D representations can be grouped into explicit and implicit categories (see Figure 2.1). Explicit representations store 3D geometric information directly, such as vertices and faces in meshes or points in point clouds. In contrast, implicit methods represent a 3D scene as a continuous function or within observations (e.g., multi-view images), where the geometry is not stored explicitly but can be recovered through function evaluation or rendered to 2D images using a corresponding rendering method.

Examples of explicit representations include RGB-D images, meshes, voxels, point clouds, and 3D Gaussian Splatting (3DGS) [90]. Meshes are widely used due to their efficiency in representing 3D surfaces and textures. Depending on mesh topology, they can be further classified into triangle meshes (tri-mesh), quadrilateral meshes (quad-mesh), etc. In this thesis, we refer to the tri-mesh representation simply as the 3D mesh. Point clouds and 3DGS can be grouped as point-based methods because both utilize points or point-like primitives to represent 3D scenes.

Examples of implicit representations include Neural Radiance Fields (NeRF) [128], Signed Distance Functions (SDF) [136], and multi-view stereo images. We classify multi-view stereo images as an implicit method because they do not store explicit geometry; instead, the 3D information is implicitly captured in multiple 2D images taken from different viewpoints. In terms of whether the 3D model is typically learned from data, 3DGS, NeRF, and SDF can also be categorized as learning-based approaches.

In this section, we focus on three specific 3D representations: 3D mesh, NeRF, and 3DGS, which form the basis of the methods developed in this thesis.

2.1.1 3D Mesh and Differentiable Rasterization

3D Mesh Basics

The 3D triangle mesh (referred to as "mesh" hereafter) utilizes a graph-like structure to represent 3D surfaces. A mesh consists of three main components: vertices, edges, and triangular faces (or facets). The basic building block of a mesh is the facet, each of which is defined by three vertices that determine its shape and 3D pose. Each edge connects a pair of vertices, and the pattern of vertex–edge connections across the entire mesh is referred to as the mesh topology. Note that vertices are typically shared by multiple facets. Because each facet has two sides, we use the ordering of its three vertices to define its orientation. Typically, the "outside" of a facet is determined using the right-hand rule in terms of the vertices' order. For example, Figure 2.2 shows a simple mesh with two facets and four vertices v_0, v_1, v_2, and v_3. One facet is formed by the ordered vertices v_0, v_2, v_1, and another facet is formed by v_1, v_2, v_3. These two facets share the vertices v_1 and v_2 and have the same orientation under the right-hand rule. In practice, a mesh is represented by its vertices and facets. Each vertex records its three-dimensional position and, optionally, its RGB color for colored meshes. Each facet typically stores the ordered indices of the three vertices that define it. The edges are implicitly recorded in the set of facets in this case.

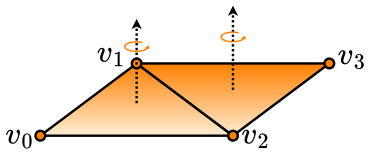


Figure 2.2: An example tri-mesh with two facets. Dotted arrows indicate the orientations of facets.

Colored Mesh and Textured Mesh

If a mesh is colored via per-vertex colors, then for each facet we use barycentric interpolation to smoothly interpolate the color within the triangle. In the example triangle facet shown in Figure 2.3, the three vertices are A, B, and C. Given a point P inside the triangle, the original triangle is conceptually divided into three sub-triangles: $\triangle PAB$, $\triangle PAC$, and $\triangle PBC$. The color at point P is computed as:

$$\text{color}(P) = a * \text{color}(A) + b * \text{color}(B) + c * \text{color}(C), \tag{2.1}$$

where a, b, and c are the weights computed by:

$$a = \frac{\text{area}(\Delta PBC)}{\text{area}(\Delta ABC)}, \quad b = \frac{\text{area}(\Delta PAC)}{\text{area}(\Delta ABC)}, \quad c = \frac{\text{area}(\Delta PAB)}{\text{area}(\Delta ABC)}. \tag{2.2}$$

The barycentric coordinates of point P with respect to the triangle $\triangle ABC$ is defined as (a, b, c).

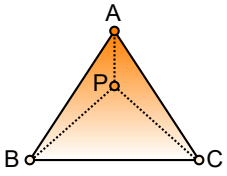


Figure 2.3: Example barycentric interpolation of vertices colors.

For a textured mesh, we typically use a UV texture map. The UV texture map is a 2D image parameterized by the (u, v) perpendicular axes. Each vertex of the 3D mesh has a corresponding coordinate in this UV space. The UV map is created

by unwrapping the 3D mesh's surface into 2D UV space. These vertex-to-UV correspondences are usually predefined and stored with the mesh data. Figure 2.4 illustrates an example 3D mesh rendered with texture and its UV texture map.

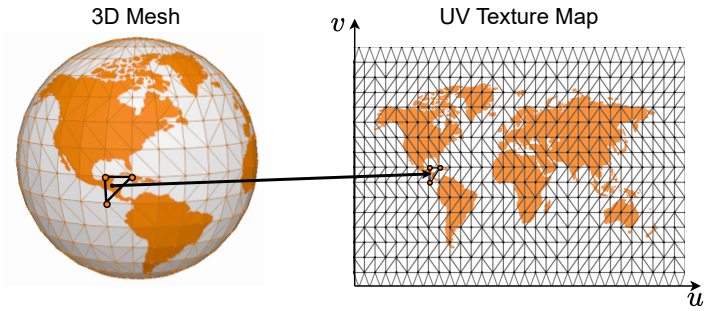


Figure 2.4: An example rendered 3D mesh and its UV texture map. The wireframes are added to visualize the mesh topology. For each triangle facet on the 3D mesh, there is a corresponding triangle on the UV map that stores its texture. We highlight an example facet and its corresponding UV texture triangle (linked by the black arrow).

Differentiable Mesh Rasterization

3D Rendering is the process of producing the 2D image(s) from a 3D scene. In 3D mesh rendering, there are two general approaches: ray-tracing-based [35], and rasterization-based [115]. Rasterization is an important step in a rasterization-based renderer; it converts the vector data (representing mathematical shapes like triangles) into a grid of pixels (a raster image) that can be displayed on a screen. Although ray-tracing methods tend to produce more photorealistic images, they are typically more computationally expensive than rasterization-based methods.

Suppose we want to use image-level supervision to learn 3D scene parameters (e.g., mesh shape, texture, lighting, and camera pose). The renderer must be differentiable so that gradients can backpropagate all the way to the scene parameters. Traditional rasterizers are not differentiable, whereas recent work has explored differentiable versions [82, 114, 115]. Figure 2.5 shows the workflow of a general differentiable rendering pipeline.

The key reasons that a traditional rasterizer is not differentiable are related to

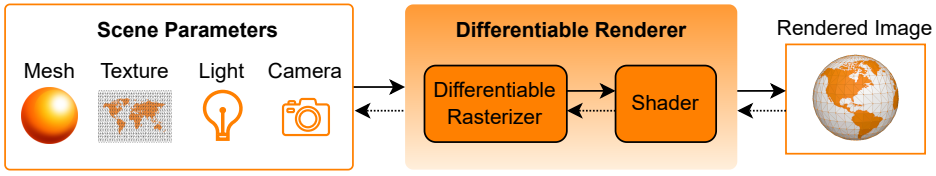


Figure 2.5: Workflow of a differentiable mesh renderer. Solid arrow indicates the forward process, and dashed arrow indicates the gradient backpropagation.

depth (z-axis) and XY discontinuities. Depth discontinuity occurs when a triangle facet becomes occluded by another facet in front of it, preventing the loss from being backpropagated to the occluded facet because visibility changes are discrete. An XY discontinuity arises when a triangle boundary crosses a pixel boundary, due to the discrete pixel grid, resulting in abrupt changes in coverage.

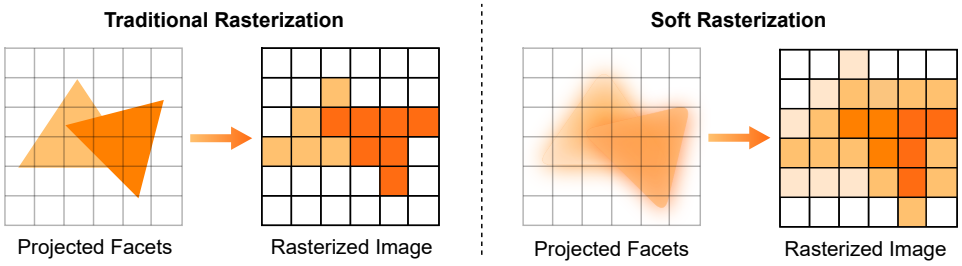


Figure 2.6: Comparison of traditional rasterization (non-differentiable) and differentiable soft rasterization.

Soft rasterization (SoftRas) addresses these challenges by using a soft, probabilistic formulation of both visibility and pixel coverage, enabling continuous gradients to flow through [114]. Figure 2.6 shows an example illustrating the behavior of rasterizing two overlapping triangle facets using the traditional method and the SoftRas method, respectively. Besides SoftRas, there are other implementations of differentiable rasterization [82, 97], but their key idea remains to address the discontinuities in the traditional rasterization process.

2.1.2 Neural Radiance Field (NeRF)

Neural Radiance Fields (NeRFs) are based on the idea that a neural network can implicitly represent a complex 3D scene [128]. The core concept is to model the 3D space as a continuous field. Rendering with NeRF can be summarized in the following steps: (1) rays are conceptually cast from the camera through the image plane into the 3D scene; (2) points are sampled along each ray, and each point encodes its spatial information (its XYZ coordinates) and the view direction; (3) the point features are passed through a neural network, typically a multi-layer perceptron (MLP), to decode their corresponding density (σ) and view-dependent color; and (4) the decoded densities and colors are aggregated along each ray using the differentiable volume rendering equation [84] to produce the final pixel color in the rendered image. The rendering is fully differentiable, and the neural network is usually trained using camera-captured ground-truth photos for image-level supervision. Figure 2.7 illustrates this process and provides a simplified visualization of how a NeRF models a continuous radiance and density field.

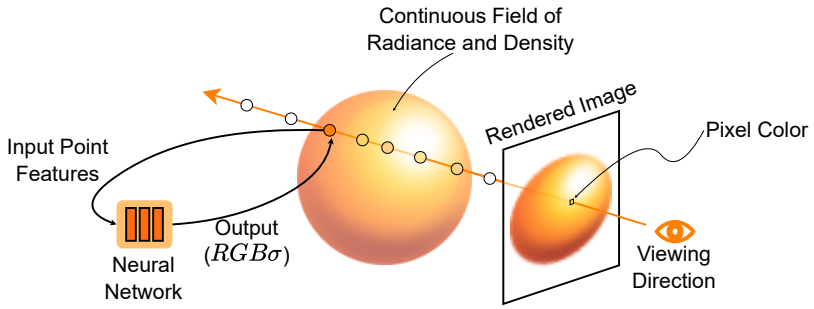


Figure 2.7: A simplified diagram demonstrating the core mechanism of a Neural Radiance Field (NeRF), showing how a 3D scene is encoded as a radiance and density field.

The volume rendering function is defined in Equation 2.3. This function assumes that the scene is represented as a continuous volumetric field that emits and absorbs light along each camera ray. A ray is parameterized as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o}$ is the camera origin, $\mathbf{d}$ is the view direction, and t is a scalar depth indicating how far a point lies along the ray. The near and far bounds of a point are denoted by t_n and t_f, such that $0 \leq t_n < t_f \ll +\infty$. Given a depth value t,

we can sample a point along the ray at $\mathbf{r}(t)$. The density at this point is denoted as $\sigma[\mathbf{r}(t)]$, and the view-dependent RGB value (radiance) is denoted as $\mathbf{c}[\mathbf{r}(t), \mathbf{d}]$. The transmittance $T(t)$ quantifies the fraction of light that travels from the point $\mathbf{r}(t)$ to the camera without being occluded. The final result $\mathbf{C}(\mathbf{r})$ is an RGB value that represents the color of the pixel corresponding to ray $\mathbf{r}$.

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma[\mathbf{r}(t)]\, \mathbf{c}[\mathbf{r}(t), \mathbf{d}]\, dt, \tag{2.3}$$

where

$$T(t) = \exp\left(- \int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds \right). \tag{2.4}$$

In practice, we need to sample a sufficient number of points along each ray to approximate $\mathbf{C}(\mathbf{r})$. Assuming that N points are sampled, the continuous volume rendering function can be approximated by its discrete version (see Equation 2.5).

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} T_i\, \alpha_i\, \mathbf{c}_i, \tag{2.5}$$

where

$$T_i = \exp\left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right), \qquad \alpha_i = 1 - \exp(-\sigma_i \delta_i). \tag{2.6}$$

Here, α_i denotes the opacity at the i^{th} sampled point. It depends on the volume density σ_i and the interval length $\delta_i = t_{i+1} - t_i$, where $t_{i+1} > t_i$. T_i is the accumulated transmittance at the i^{th} sampled point.

2.1.3 3D Gaussian Splatting (3DGS)

The 3D Gaussian Splatting (3DGS) technique presents a 3D Gaussian primitive–based scene representation along with a splatting-based rendering process, enabling differentiable and efficient rendering [90]. Each 3D Gaussian primitive is modeled as an anisotropic, unnormalized three-dimensional Gaussian kernel. This kernel defines the spatial influence of a 3D Gaussian $\mathbf{g}_i$ on any 3D location $\mathbf{x} \in \mathbb{R}^3$:

$$G_i(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1} (\mathbf{x}-\mu_i)}, \tag{2.7}$$

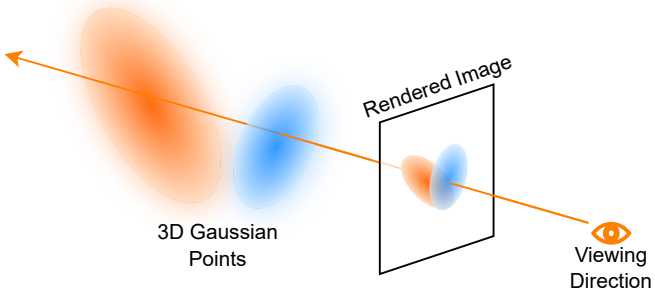


Figure 2.8: An example illustrating the 3D Gaussian Splatting (3DGS) process. In this example, two 3D Gaussians are projected onto the 2D image plane to compute the corresponding pixel colors. Note that this figure focuses solely on the rendering of 3D Gaussians and omits the initialization and training components of a full 3DGS pipeline.

where $\mu_i \in \mathbb{R}^3$ is the mean (center of the 3D Gaussian) and $\Sigma_i \in \mathbb{R}^{3\times3}$ is the covariance matrix. To ensure the covariance matrix Σ_i to be positive semi-definite, 3DGS computes the Σ_i based on a scaling matrix $S_i \in \mathbb{R}^{3\times3}$ and a rotation matrix $R_i \in \mathbb{R}^{3\times3}$:

$$\Sigma_i = R_i S_i S_i^\top R_i^\top . \tag{2.8}$$

This representation is analogous to defining the scaling and rotation of an ellipsoid [90]. Further, the scaling and rotation matrices can be derived from a scaling vector $\mathbf{s}_i \in \mathbb{R}^3$ and a quaternion $\mathbf{q}_i \in \mathbb{R}^4$. Employing $\mathbf{q}_i$ rather than directly using rotation angles helps to circumvent the Gimbal lock issue.

The 3D Gaussian $\mathbf{g}_i$ is then parameterized by $\mathbf{g}_i = \{\mu_i, \mathbf{s}_i, \mathbf{q}_i, \alpha_i, \mathbf{a}_i\}$, where α_i is the opacity analogous to NeRF's density, and $\mathbf{a}_i$ is the vector of Spherical Harmonic (SH) coefficients for computing the view-dependent RGB color [50].

The 3D Gaussians are projected to 2D for rendering (namely splatting). The image formation model of 3DGS mirrors the volume rendering techniques utilized in NeRF [43, 128]:

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

where p is the 2D-pixel location on the plane onto which 3D Gaussians are pro-

jected. Here, $\mathbf{C}(p)$ represents the final color of the pixel, $\mathcal{N}_p$ is an ordered (front-to-back) set of the 2D projections of the 3D Gaussians that cover the pixel, and $\mathbf{c}_i^p$ is the computed color of the i^{th} 2D projection at location p [226]. Figure 2.8 illustrates the rendering process of 3DGS via a simple example. We do not cover the full 3DGS training pipeline in this section.

Although both NeRF and 3DGS rely on volume rendering, they differ fundamentally in their underlying 3D representations. NeRF represents a continuous radiance field implicitly via a neural network, whereas 3DGS approximates it explicitly using a set of Gaussian primitives. Therefore, NeRF has a smooth and continuous representation but requires dense ray sampling and neural network inference during rendering, making it computationally expensive. In contrast, 3DGS achieves real-time rendering by replacing the implicit field with a finite set of explicit 3D Gaussians that directly store color and opacity, enabling fast rendering.

2.2 3D Morphable Face/Head Models (3DMMs)

The 3D morphable face model (3DMM) [14] uses the 3D mesh structure for the 3D face representation. Since the scanned 3D faces are registered to a common mesh topology, the 3D surface shape of a registered 3D face can be represented as a matrix $S \in \mathbb{R}^{V\times 3}$ of 3D mesh vertex locations, where V is the total number of vertices. The shape of a face can be synthesized through the following shape model formula [109]:

$$S_{model} = S_{mean} + c_{id}\mathbf{S}_{id} + c_{expr}\mathbf{S}_{expr}, \tag{2.10}$$

where S_{mean} is the average face shape across the dataset, $\mathbf{S}_{id} \in \mathbb{R}^{P_{id}\times V\times 3}$ and $\mathbf{S}_{expr} \in \mathbb{R}^{P_{expr}\times V\times 3}$ are the principal components of identity and expression residuals derived from the dataset, $c_{id} \in \mathbb{R}^{P_{id}}$ and $c_{expr} \in \mathbb{R}^{P_{expr}}$ represent the corresponding coefficients that control the synthesized face shape, P_{id} and P_{expr} indicate the numbers of identity and expression components respectively.

The original 3DMM uses mesh vertex colors to represent the texture of a 3D face [139]. Similar to the shape model, the texture model is formulated as:

$$T_{model} = T_{mean} + c_{tex}\mathbf{T}, \tag{2.11}$$

where $T_{mean} \in \mathbb{R}^{V\times 3}$ are the average vertex colors derived from the dataset, $\mathbf{T} \in \mathbb{R}^{P_{tex}\times N\times 3}$ are the principal components of the vertex color residuals, and $c_{tex} \in \mathbb{R}^{P_{tex}}$ represent the texture coefficients. Note that the vertex colors do not include shading, shadows, or other lighting effects, and are therefore often referred to as albedo. The lighting effects are approximated by the SH lighting [167], with coefficients c_{light} [38, 48].

An important application of 3DMMs is the reconstruction of the 3D face from a single-image. Existing approaches can be broadly categorized into fitting-based methods [14, 246] and learning-based methods [38, 47, 95, 186]. Both types of methods rely on a 3DMM to provide strong prior knowledge of plausible human face geometry and appearance. The primary difference lies in how the 3DMM parameters are estimated. Fitting-based methods optimize 3DMM parameters directly by iteratively minimizing the discrepancy between the rendered 2D image of the reconstructed 3D face and the target face image. In contrast, learning-based methods train a parametric regressor (typically a neural network) to predict the 3DMM parameters given the input image in a single forward pass. The training of such networks commonly relies on ground-truth 3D scans and/or input images to provide supervision.

There are many variants of 3DMMs [16, 47, 109, 139, 142, 186]. Some works replace the PCA-based formulation with neural network–based models [47, 186]. More recent models, such as FLAME [109] and UHM [142], extend the traditional 3DMM beyond the facial region to represent the entire bare head (excluding facial hair), and additionally incorporate anatomical structures such as the eyeballs and neck; UHM further includes the oral cavity and teeth. To support more expressive and anatomically plausible facial dynamics, FLAME employs Linear Blend Skinning (LBS) to model articulated jaw and neck rotations, which enables more flexible and exaggerated facial expressions.

2.3 Deep Generative Modeling

Deep generative modeling represents a class of machine learning approaches that use deep neural networks to learn the underlying patterns of a dataset and to synthesize new data that resembles the original [163, 209]. A generative model maps

latent variables (its input parameters) sampled from a prior distribution (e.g., a Gaussian distribution) to generate synthetic samples. The generative model approximates the real data distribution through a learned transformation. Let the generative model be denoted as a function $F_\theta(\alpha)$, where θ denotes the model parameters and α denotes the input latent variables. A sampled data point $\hat{x}$ is obtained as:

$$F_\theta(\alpha) = \hat{x}, \tag{2.12}$$

where $\alpha \sim p(\alpha)$ means the latent variables are sampled from the prior distribution $p(\alpha)$. The generated data can be viewed as a sample drawn from the model's learned distribution, i.e., $\hat{x} \sim p_\theta(x)$. The goal of generative modeling is to learn a model distribution $p_\theta(x)$ to approximate the true data distribution $p_{\text{data}}(x)$, i.e., $p_\theta(x) \approx p_{\text{data}}(x)$.

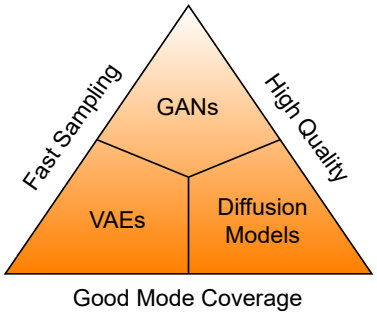


Figure 2.9: Characteristics (sampling speed, generation quality, and mode coverage) of the three important generative modeling methods in computer vision: Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and diffusion models.

There are three important deep generative modeling methods in computer vision: Variational Autoencoders (VAEs) [92], Generative Adversarial Networks (GANs) [63], and diffusion models [70]. Figure 2.9 shows the characteristics of these methods. Among these three categories, VAEs and GANs are the most relevant to this thesis because they support fast sampling, which is important in reducing the cost of 3D face/head modeling. In contrast, although diffusion models have recently achieved state-of-the-art performance in image and 3D content generation, they are computationally more expensive in sampling and training [193].

In the following, we briefly review VAEs, GANs, Diffusion Models, as well as the Autodecoders (ADs). It is worth noting that ADs are not generative models in the traditional sense but can be adapted to perform generative modeling when combined with a suitable latent prior.

2.3.1 Variational Autoencoder

It is helpful first to review a standard Autoencoder (AE) to understand the Variational Autoencoder (VAE). An Autoencoder is a neural network model consisting of an encoder and a decoder. The encoder maps an input data point x to a latent code z, and the decoder reconstructs the data by mapping z back to $\hat{x}$ (see Figure 2.10). An AE is trained in a self-supervised way using a reconstruction loss. A trained AE is usually used to extract low-dimensional features (latent codes). However, AEs are not effective generative models because their latent code distribution does not follow any predefined prior, making it impossible to directly sample meaningful latent vectors z for generating new data. In addition, AEs are deterministic models that map each input to a single point in the latent space.

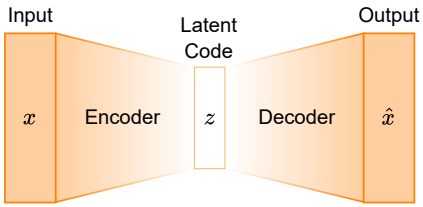


Figure 2.10: Illustration of an Autoencoder (AE).

Different from the AE, the Variational Autoencoder (VAE) is a probabilistic model. VAE addresses the limitation of the AE by imposing a prior distribution, typically a normal distribution, over the latent space [92]. This design enables both the reconstruction ability from AE and the sampling ability for data generation (see Figure 2.11). The encoder of a VAE maps an input data point x to the parameters of a probability distribution over a latent code $z \in \mathbb{R}^k$, where k is the latent code dimension. Specifically, the encoder's outputs are two vectors: the mean $\mu \in \mathbb{R}^k$ and the standard deviation $\sigma \in \mathbb{R}^k$. We assume a unit diagonal Gaussian prior on the latent variables, $p_\theta(z) = \mathcal{N}(0, I)$, where θ indicates the generator model

parameters. The use of the Gaussian distribution is mainly because (1) it is simple and easy to sample from, and (2) it regularizes the latent space to be continuous and smooth, which is crucial for generative tasks. During the forward pass, a latent code z is sampled based on μ and σ, and the decoder (also the generator) reconstructs the original input via the likelihood model $p_\theta(x|z)$.

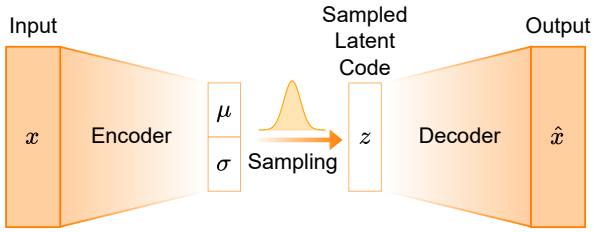


Figure 2.11: Illustration of a Variational Autoencoder (VAE).

One objective of a VAE is to approximate the true posterior $p_\theta(z|x)$ in order to learn meaningful latent representations and enable data generation. Because the true posterior is intractable, the encoder introduces a variational distribution $q_\phi(z|x)$, parameterized by ϕ. We optimize $q_\phi(z|x)$ to be close to $p_\theta(z|x)$. Ideally, this will be achieved by minimizing the following Kullback-Leibler (KL) divergence:

$$\mathrm{KL}\big(q_\phi(z|x) \,\|\, p_\theta(z|x)\big) = \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z|x) - \log p_\theta(z|x)\right]. \tag{2.13}$$

However, Equation 2.13 cannot be computed directly. Instead, the VAE maximizes the Evidence Lower Bound (ELBO), which learns $q_\phi(z|x)$ to approximate $p_\theta(z|x)$ indirectly. In practice, this is implemented by minimizing a loss function including a reconstruction term and a KL divergence regularization term:

$$\mathcal{L} = \underbrace{-\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{reconstruction loss}} + \underbrace{\mathrm{KL}(q_\phi(z|x)||p_\theta(z))}_{\text{KL divergence loss}}. \tag{2.14}$$

Because sampling from a normal distribution is not differentiable, the VAE uses the reparameterization trick to allow gradients to flow through μ and σ back to the encoder parameters. The trick is based on the fact that a Gaussian random

variable can be expressed as:

$$z = \mu + \epsilon \odot \sigma, \epsilon \sim \mathcal{N}(0, I), \tag{2.15}$$

which is equivalent to sampling $z \sim \mathcal{N}(\mu, \text{diag}(\sigma^2))$. In this formulation, the source of randomness is ϵ, which is independent of the model parameters and does not require gradients. Meanwhile, μ and σ in this formulation remain fully differentiable, so that the VAE model can be trained with gradient-based optimization.

2.3.2 Autodecoder with Prior

The concept of an autodecoder (AD) dates back to the work of Tan et al. [178], where a neural network decoder and the latent codes of the samples are jointly optimized for dimensionality reduction. Figure 2.12 illustrates the autodecoder structure, which is similar to that of an autoencoder but without the encoder component. The traditional AD remains a deterministic model, with a latent space that is typically irregular and prone to overfitting to the training data.

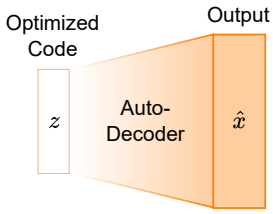


Figure 2.12: Illustration of an Autodecoder (AD).

To enable data generation, Park et al. [136] proposed a probabilistic formulation of the AD. By assuming a zero-mean isotropic Gaussian prior over the latent codes, $p(z) = \mathcal{N}(0, \sigma^2 I)$, the optimization objective becomes minimizing the negative log-posterior. This assumption leads to a loss function consisting of a reconstruction term and a regularization term derived from the prior:

$$\mathcal{L} = \mathcal{L}_{recon} + \frac{1}{\sigma^2}\|z\|_2^2. \tag{2.16}$$

This L_2 regularization encourages the latent codes to follow the assumed Gaussian prior, resulting in a smooth and continuous latent space. As a result, new data can

be generated by sampling z from the prior distribution.

2.3.3 Generative Adversarial Network

The generative adversarial learning framework proposed by Goodfellow et al. approaches the estimation of generative models through an adversarial process [63]. This framework is inspired by a minimax two-player game: a generator network (G) and a discriminator network (D) are trained to compete with each other. Therefore, the framework is also known as the Generative Adversarial Network (GAN). The discriminator acts as a binary classifier. Its objective is to distinguish real data samples x from generated samples $\hat{x}$ by the generator. The generator takes as input random noise vectors sampled from a prior distribution, $z \sim p(z)$ (typically a Gaussian distribution), and aims to produce generated samples that fool the discriminator. Ideally, after training, the generator learns to approximate the actual data distribution. At the same time, the discriminator outputs probabilities close to 0.5 for both real and generated samples. That is to say, the discriminator can no longer reliably distinguish between real and fake samples. Figure 2.13 illustrates the GAN framework.

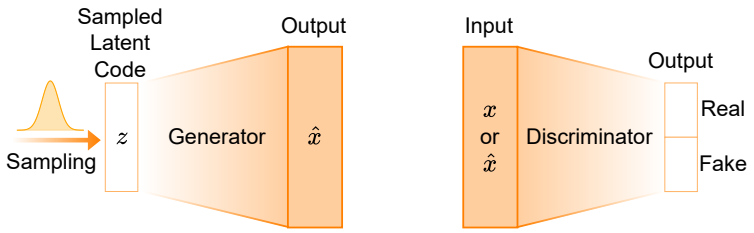


Figure 2.13: Illustration of a Generative Adversarial Network (GAN).

The objective of GAN training can be formulated as:

$$\min_G \max_D [\mathbb{E}_{x\sim p_{data}(x)} \log D(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D(\hat{x}))], \hat{x} = G(z). \quad (2.17)$$

In practice, the generator and discriminator are optimized alternately to ensure stable training. The loss function used to train the discriminator is defined as

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}(x)} \log D(x) - \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]. \quad (2.18)$$

During generator training, the real data distribution guides the learning implicitly through the discriminator. The generator is updated using the following loss function:

$$\mathcal{L}_G = \mathbb{E}_{z \sim p(z)} \log(1 - D(G(z))). \tag{2.19}$$

Unlike VAEs, GANs estimate the data distribution implicitly. VAEs optimize the ELBO, which often leads to blurry outputs (e.g., over-smoothed images). In contrast, the discriminator in a GAN penalizes unrealistic samples, which helps the generator to capture high-frequency details. However, GAN training is not a simple optimization problem. It corresponds to finding an equilibrium between the generator and the discriminator. Therefore, GANs are prone to instability during training. Common issues include mode collapse, vanishing gradients, and failure to converge.

2.3.4 Diffusion Model

The concept of diffusion models was proposed in [168] and made practical in Denoising Diffusion Probabilistic Models (DDPM) [70]. The main idea of diffusion models is to model the data generation process as recovering the data from noise. Define x_0 as the original data point (e.g., an image); the process of adding random noise to it over T steps is known as the forward process in diffusion models: $q(x_t \mid x_{t-1})$, where $t \in \{1, \ldots, T\}$. In DDPM, the forward process does not require learning. The reverse process recovers the data step-by-step, formulated as $p_\theta(x_{t-1} \mid x_t)$, using a neural network parameterized by θ. Figure 2.14 shows the Markov chain of a DDPM.

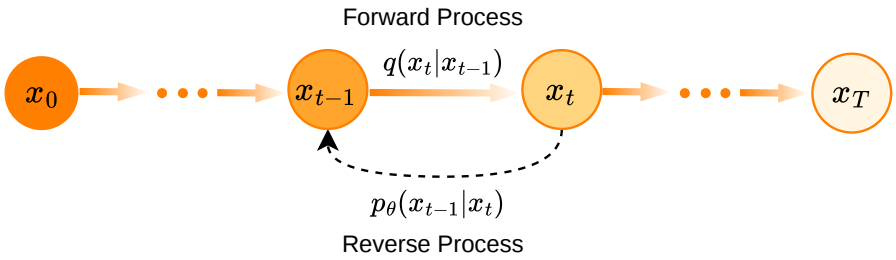


Figure 2.14: Illustration of a diffusion model (adapted from [70]).

Diffusion models are now widely used in image, video, and 3D asset generation tasks [152, 193, 213, 232]. In head avatar generation, one approach uses 2D diffusion models to guide 3D head avatar generation. These methods can be further grouped into those that generate multi-view training data to train a 3DGS-based head avatar model [23, 180], and those that leverage 2D diffusion models through Score Distillation Sampling (SDS) [58, 143]. The other approach directly generates 2D video frames of the avatar [79, 175]. In general, diffusion-based head avatar methods are more computationally expensive than traditional approaches. Nevertheless, recent diffusion-based avatar methods produce photorealistic, geometrically consistent, and expressive results. While this thesis does not employ diffusion models, focusing instead on more efficient approaches, the strong generative capability of diffusion models suggests a promising direction for future extensions of this work.

Chapter 3

Mesh-Based 3D Face Creation

I thought it wasn't real. — Neo, The Matrix (1999)

3.1 Introduction

It has become popular for individuals recently to allow their faces to be used in 3D virtual reality (VR) applications and user-generated content games. For example, in a virtual conference or a virtual get-together meeting, the use of reconstructed 3D faces that resemble those of the real users helps the participants to identify the persons behind the avatars. The reconstruction of 3D faces is based predominantly on the 3D morphable face model (3DMM) built from 3D scanned human faces [45]. While 3D face reconstruction has been well studied in the past two decades [7, 14, 38, 45, 95, 113, 186, 245, 246, 250], two closely related tasks, i.e., the editing and stylization of reconstructed 3D faces, have remained relatively underexplored. We believe there is great potential for deep learning-based 3D face editing and stylization in many real-world applications, including 3D gaming, filmmaking, VR, and plastic surgery. 3D face editing can also benefit 2D face editing because the 3D face can provide explicit geometry information to 2D generative models [39, 141, 206].

Enabling a user to edit their reconstructed 3D face can solve a main problem in existing 3D face reconstruction methods. As the reconstructed 3D face is some-

times not accurate enough, the user may want to correct it. Also, when the user is not satisfied with some parts of their actual face, they can modify those parts. However, 3D editing is a challenging task. Editing a 3D shape usually requires strong spatial thinking ability, and traditional 3D editing tools are designed for use by artists. Moreover, editing of 3D faces is even more difficult than editing many other 3D shapes, because people are very sensitive to the appearance of a human face (minor modifications could cause very different feelings in people). Because facial parts such as eyebrows and nose are relatively independent, segmenting the 3D face into parts and independently editing each part makes 3D face editing easier to handle [49, 59, 181]. In this thesis, we want to take one step further to bridge single-image 3D face reconstruction (SIFR) and 3D face editing. Since ambiguities such as depth and lighting will cause the reconstructed 3D face to be different from the actual face, SIFR is an ill-posed problem. Recent works on SIFR display limited improvement in reconstruction when compared to the 3D ground-truth [38, 48], whereas 3D-to-2D alignment, i.e. aligning the 3D face with the image it was reconstructed from, is still an open challenge [123]. Therefore, we also explore improving the 3D-to-2D alignment by taking advantage of 3D face editing. Inspired by [95, 246], we use 2D face semantic segmentation/parsing to provide ground truth in aligning the 3D face to the image. This requires us to segment our 3D face model based on the 2D face parsing, which is different from how a 3D face is segmented in [49, 59, 181].

Beyond editing, 3D stylization of reconstructed faces has also become an increasingly important research topic [67, 225, 229], driven by the demand for user-customized 3D avatars in virtual reality and video games. Such stylized faces are usually handcrafted by 3D artists, making the creation of distinct morphable models for each style costly. We propose a novel approach to the shape stylization problem and are the first to propose using a single artist-designed template for each face style. The main idea is to model the 3D face stylization task as that of transferring face features in the source 3D face (e.g., nose or mouth shapes) to a style template face designed by an artist, such that the resulting stylized face preserves the face features from the original face (source) as well as the style from the template (target). A significant advantage here is that we only need the 3D artist to design one template for each style. In contrast to existing methods that exclusively

focus on generating facial expressions for style templates [135, 197, 229, 241], our approach solves the transfer of facial features (such as nose shape) from any reconstructed face to the style templates, thereby allowing greater diversity in identity representation within each style.

In this chapter, for the 3D mesh-based face representation, we advance existing works by proposing two data-driven methods to realize 3D face reconstruction, editing, and stylization. Our aim is to achieve greater controllability over the constructed face, such as modifying some parts or editing the reconstructed 3D face. Our first method, NEO-3DF (Section 3.3), addresses single-image-based 3D face reconstruction and further editing within a single framework. The key to achieving intuitive 3D face editing lies in the proposed face parts-based model. Most SIFR methods use a global face model (i.e., the 3D face shape is modeled as a whole), which makes it extremely challenging for editing [38, 48, 103]. Our 3D face modeling, however, uses different sub-models for the different semantic segments (parts) of the face. This ensures that editing the shape of one part of a face does not cause changes in other parts, resulting in easier and more convenient editing. This approach also allows us to improve the 3D reconstruction accuracy using 2D supervision. We developed a way of using the As-Rigid-As-Possible (ARAP) method [170] to blend the face segments, so the face appears natural at the segment boundaries. Importantly, we propose a differentiable version of ARAP that makes it possible to use gradient descent-based optimization to improve the 3D-to-2D alignment between the reconstructed face segments and those of the original 2D image. In the second part of this chapter (Section 3.4), we study face stylization. We introduce a method based on a weakly supervised learning approach to transfer facial features from a reconstructed 3D face to an artist-designed 3D stylized face template. Inspired by traditional deformation transfer methods, our network first builds vertex-level correspondences between the source and target meshes, then naively maps the vertex-level deformation (vertex offsets) from the source to the target mesh. To address the noise in the mapped offsets, we propose a template mesh-guided smoothing loss based on the Laplacian mesh smoothing filter [132] but guided by the high-quality template mesh instead of the noisy mesh, and we further employ As-Rigid-As-Possible (ARAP) regularization [75] to preserve the mesh's local rigidity during training. Together, these two methods enable both the

editing (the first method) and stylization (the second method) of a reconstructed 3D face.

3.2 Related Works

3.2.1 3D Face Modeling and Reconstruction

The 3D face is usually represented by the 3D mesh structure, where the vertices and facets define the surface of the 3D face. A straightforward way to create a 3D face is by using a weighted average of a linear combination of a set of example 3D faces [191]. An improved version of this approach, the 3D Morphable Face Model (3DMM), transforms the scanned and processed example 3D faces to a vector space representation, using Principal Component Analysis (PCA) [14]. The PCA captures the primary modes of variation in the pre-collected 3D face dataset. Although 3DMM was proposed two decades ago, it is still widely used in current human face-related research and applications, such as 3D face reconstruction [111, 246], controllable face image generation [39, 206], and 3D face animation [7]. In recent years, research on the use of deep learning to learn the general 3D face model shows the advantage of neural network-based 3D face models over traditional 3DMMs [18, 148, 186].

Single-image 3D face reconstruction (SIFR) is an important application of both 3DMMs and neural network-based 3D face models [45]. The task is to reconstruct a 3D face from a given single-view face image. In most cases, the camera's intrinsic and lighting conditions are unknown. In addition, for images where the faces appear identical, the actual 3D shape of these faces can be different due to potential ambiguities. Thus, the SIFR is an ill-posed problem, and we can only approximately reconstruct a 3D face from a single image. However, compared with accurate 3D reconstruction methods, using expensive 3D scanning devices or multi-view images, SIFR is more accessible to users of consumer electronics. Existing SIFR approaches can be divided into two main paradigms: *fitting-based* and *learning-based* methods. Fitting-based methods, also known as analysis-by-synthesis, estimate the parameters of a 3DMM by iteratively aligning a rendered synthetic face with the input image [14, 246]. In contrast, learning-based methods

employ a neural network to directly regress the 3DMM parameters from the input image [38, 47, 95, 186]. Such models are typically trained using ground-truth 3D scans and/or image-space-based supervision.

3.2.2 3D Face Stylization

3D stylization of reconstructed faces has become an increasingly important research topic [67, 225, 229], driven by the demand for user-customized 3D avatars in virtual reality and video games. Existing methods can be broadly grouped by their input modality. One line of work focuses on generating facial expressions for style templates from 3D inputs [135, 197, 229, 241], which provides expression diversity within each style but does not address the transfer of identity-related facial features across styles. Another line of work generates 3D stylized faces directly from 2D images [78, 125, 140, 194, 208, 231]. However, these 2D-to-3D approaches have several limitations. Some rely on neural rendering, where the 3D representation is not mesh-based, making facial animation control difficult and limiting compatibility with mainstream mesh-based applications [125, 208, 231]. Toonify3D [78] stylizes the face in 2D image space using StyleGAN and then applies non-rigid registration to a template 3D head mesh; however, this method requires fine-tuning a StyleGAN model for each style using a large number of images, adding significant computational overhead. Overall, while these methods do not require 3D training data, they still depend on a substantial amount of 2D images, which introduces additional constraints. In contrast, the second method proposed in this chapter (Section 3.4) takes a different approach, focusing on deformation-transfer-based stylization that requires only a single artist-designed 3D template per style and allows the transfer of facial features (such as nose shape) from any reconstructed face to the style templates, thereby allowing greater diversity in identity representation within each style.

3.3 Method 1: Editing-Oriented 3D Face Reconstruction (NEO-3DF)

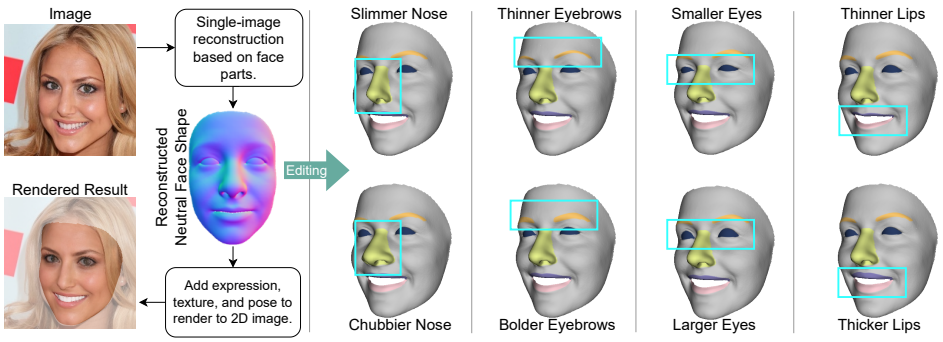


Figure 3.1: Demonstration of our intuitive face editing results. The left side displays the face image (top), the reconstructed neutral 3D face shape, and the rendered result (bottom). The other images are examples of editing different face parts.

3.3.1 Overview

Presently, acquiring 2D face images (e.g., by a conventional camera) is easy, but acquiring a 3D face is not. Existing methods, therefore, create a 3D face from a 2D face image (referred to as single image-based 3D face reconstruction). A user may wish to edit the reconstructed 3D face, but 3D face editing has seldom studied before. This section presents such an editing method and shows that reconstruction and editing can help each other. In the proposed framework below, which we name NEO-3DF, the proposed 3D face model has independent sub-models corresponding to different semantic parts of a face. NEO-3DF allows us to achieve both local intuitive editing and better 3D-to-2D alignment (between the 3D reconstructed face and the 2D image from which it is reconstructed). Each face part in our model has a set of controllers designed to allow users to edit the features in the corresponding face part (e.g., nose height in the nose part of the face). In addition, we propose a differentiable module for blending the adjacent face parts and enabling the automatic adjustment of the face parts (both the shapes and the locations) so that they are better aligned with the original 2D image. Experiments show

that NEO-3DF outperforms existing methods in intuitive face editing and has better 3D-to-2D alignment accuracy (14% higher IoU) than global face model-based reconstruction.

In summary, our main contributions are:

1. We proposed NEO-3DF, the first method that couples single-image 3D face reconstruction and intuitive 3D face editing;

2. The NEO-3DF is face-parts-based and follows 2D semantic face segmentation, which makes it possible for an automatic shape adjusting process;

3. We proposed a differentiable version of ARAP to allow us to use 2D face segmentation as guidance to adjust the reconstructed 3D face shape automatically;

4. We associated editing controllers with the latent space and justified the effectiveness of its use in an automatic 3D-to-2D alignment process.

3.3.2 Background

General-Purpose 3D Mesh Editing

Many 3D mesh editing methods allow the user to modify the location of some vertices (constraint vertices) to edit the 3D mesh. The cage-based deformation (CBD) method warps the space enclosed by "cages" to deform the mesh surface [100]. CBD is usually used for large-scale deformations such as human body postures and is not ideal for 3D face editing because the latter focuses more on local details. Building the appropriate "cages" is also a challenging task. The ARAP method is a detail-preserving mesh surface deformation method [170]. ARAP assumes that by making the local surface deformation close to rigid, one can preserve the local surface details. In editing 3D face shapes, neither CBD nor ARAP can prevent improper edits, this is because these methods do not have prior knowledge on 3D faces. In comparison, the blendshape methods solve the face shape editing problem by using a set of pre-defined shape offsets to deform a given face shape [104]. Although the concept of the blendshape is straightforward, developing intuitive

meaningful blendshapes is labor-intensive. 3DMMs can be considered as a special type of blendshape method, where the blendshapes are derived from 3D scans and compressed by PCA. However, the eigenvectors do not have intuitive meaning.

3D Face Editing

In the video gaming industry, 3D face editing systems are usually designed by artists. For example, the artists can define meaningful controllers (e.g., nose height) with corresponding handcrafted blendshapes to allow users to easily change the shape of a 3D face [161]. Foti et al. proposed the Latent Disentanglement Variational Autoencoder (LD-VAE) and demonstrated that it can be used for constraint vertex-based face shape editing [49]. Because latent disentanglement in terms of the face parts is an objective during the training of LD-VAE, the local editing result is better than using a traditional 3DMM. Whereas this type of editing still requires the user to have some skills in arts. Ghafourzadeh et al. developed a face editing system that uses a face parts-based 3DMM (PB-3DMM) [59]. This method decomposes the face shape into five manually selected non-overlapping parts to gain local control. The models of these face parts are built using PCA on 135 3D face scans. They also use the method proposed by Allen et al. to find a linear mapping between anthropometric measurement space and the PCA coefficient spaces to achieve intuitive control/editing (such as changing the nose height) [2]. PB-3DMM is the first data-driven 3D face model that supports intuitive local editing. However, like traditional 3DMM [14], the PB-3DMM still relies on 3D scans, which are expensive to collect and process. Moreover, PB-3DMM uses a specially designed low-polygon mesh topology and is not optimized for single-image 3D face reconstruction and 3D-to-2D face alignment. We briefly summarize some important factors of PB-3DMM, LD-VAE and the proposed NEO-3DF in Table 3.1.

3.3.3 The Proposed Method

Overview

The proposed framework (see Figure 3.2) supports single-view 3D face reconstruction and its further editing. In this work, we first reconstruct a neutral 3D face

Table 3.1: Comparison of the NEO-3DF and other parts-based 3D face methods.

Method	Model	Mesh Info. (Vertices)	Face Parsing	Intuitive Editing	Parts-Blending	Back-prop.	SIFR
PB-3DMM [59]	multiple PCAs	specially designed (6,014)	✗	✓	✓	✗	✗
LD-VAE [49]	single VAE	FLAME [109] (71,928)	✗	✗	N/A	✓	✗
NEO-3DF (proposed)	multiple VAEs	modified BFM [38, 139] (35,709)	✓	✓	✓	✓	✓

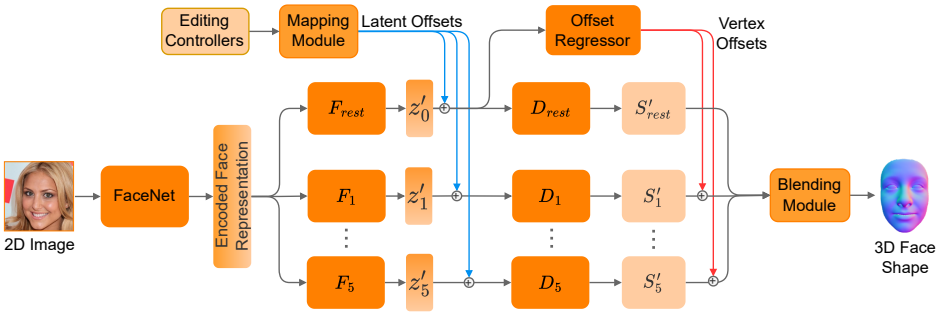


Figure 3.2: Flowchart of the proposed NEO-3DF. FaceNet: The convolutional neural network encodes the 2D face image to a vector representation. Networks F_i estimate latent representations z'_i of corresponding face segments, and shape decoders D_i reconstruct the segment shape S'_i from z'_i. The Offset Regressor predicts offsets used to assemble the segment shapes to their correct location.

shape. The expression and texture of the 3D face can be added later through existing methods (such as [26] and [111]). We use a pre-trained FaceNet [155] as the face image encoder. We segment our 3D face topology into six segments (parts). Each segment has its own editing controllers. These controllers determine the variables that can be changed in this segment (e.g., nose height). For each segment i, we have a regressor network F_i that estimates the latent encoding z'_i of the segment. Following each encoder F_i is the shape decoder D_i, which reconstructs the shape S'_i. The set of editing controllers enables intuitive face shape editing. To find the mapping between the editing controllers space and the neural network latent

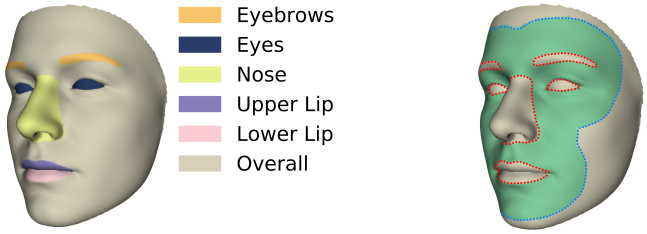


Figure 3.3: Left: The six face segments considered. Right: Shape boundaries (semantic segment boundaries in red lines, outer boundary in blue line) and transition area (in green).

space, we use a similar way as used in [2, 59]. The main idea is to construct a linear system representing the mapping function and solve the unknown parameters of the mapping function. The blending module uses ARAP optimization to derive an overall natural-looking face shape.

Semantic Face-Segment Based 3D Face Model

To develop a face model based on semantic face-segment, we use the Basel Face Model (BFM) [139] as the base model. We consider the face to have six segments: five semantic face segments (*eyebrows*, *eyes*, *nose*, *upper* and *lower lips*) and one segment for the *rest* of the face. Following [38], we remove the neck and ears from the original BFM mesh topology. The resulting 3D face mesh has 35,709 vertices. To automatically segment the 3D face, we use a similar approach as in [95]. The main idea is to render the randomly synthesized 3D faces to 2D, then run the 2D face parsing model to get 2D parsing labels. Finally, we can map the 2D pixel-level labels back to 3D vertices. This automatic 3D face segmentation process ensures the 3D vertex labels are consistent with the mainstream face image semantic parsing, which is essential in our following work on fine-tuning the model, using ground-truth 2D face parsing masks as supervision. Figure 3.3 (left) shows the face parts of our 3D face mesh topology.

We denote the 3D shape S as a set of vertices $S = \{v_i | v_i \in \mathbb{R}^3\}_{i \in V}$, where V is the set of vertex indices of the shape S, v_i represents the 3D location of the i^{th} vertex in S. We use $S_k (k \in \{1..5\})$ to represent the five semantic face segments of eyebrows, eyes, nose, upper lip, and lower lip respectively. For convenience, we

call S_ks the five-segments. The shape of the rest segment S_{rest} is a little different from S_ks, because it has a one-ring overlap with each of the S_ks. This one-ring overlap is important in our ARAP-based blending module which provides the curvature information at the segment boundaries (see Figure 3.3).

We use Multi-Layer Perceptron (MLP) based Variational Autoencoders (VAEs) to learn the 3D shape representation. In total, we train six VAEs for each face segment i (E_i and D_i are encoder and decoder respectively). For S_{rest}, the latent representation is a vector $z_0 \in \mathbb{R}^{1\times 30}$, and for S_ks, $z_k \in \mathbb{R}^{1\times 10}$. We pre-process each segment shape in the dataset, except S_{rest}, to make the geometric center of the dataset's 3D bounding box as the origin of the 3D Cartesian coordinate system. The reason is that we expect the shape decoders of the five-segments to only care about the shape itself rather than the relative location of the segment on the face. The loss in training the VAEs is defined as:

$$\mathcal{L}_{VAE} = \lambda_{recon}\mathcal{L}_{recon} + \lambda_{KL}\mathcal{L}_{KL} + \lambda_{smooth}\mathcal{L}_{smooth}, \tag{3.1}$$

where $\mathcal{L}_{recon}$ is the shape reconstruction loss, $\mathcal{L}_{KL}$ is the Kullback-Leibler (KL) divergence loss, $\mathcal{L}_{smooth}$ is the cotangent-weight Laplacian loss, and the λs are loss term weights. Denote v and v' as the vertices of the ground-truth shape S and reconstructed shape S' respectively. We compute the reconstruction loss $\mathcal{L}_{recon}$ as $\sum_{i\in V} \|v_i - v'_i\|_2^2$. The $\mathcal{L}_{KL}$ is the KL divergence of the distribution of $z \in \mathbb{R}^{1\times d_z}$ from a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I} \in \mathbb{R}^{d_z\times d_z}$ is the identity matrix and $\mathbf{0} \in \mathbb{R}^{1\times d_z}$ is a vector of zeros. The Laplacian loss $\mathcal{L}_{smooth}$ helps the decoder to generate smooth mesh surfaces.

Because the input to E_i is the 3D shape, we still need to couple the 2D face image encoding with the 3D shape decoders D_i. To do this, after VAE training we discard the VAE encoders E_i and add another set of regressor networks F_i between the 2D face image encoding and the 3D shape decoders D_i. To pre-train F_i, we freeze the parameters of the face image encoder and shape decoders. The loss function is the same as (3.1) except for discarding the $\mathcal{L}_{smooth}$ term. Here, $\mathcal{L}_{KL}$ acts as a regularization term to prevent over-fitting.

Neural Network Architectures

The face image encoder in our framework is a FaceNet [155]. The network architecture is Inception-v1 [176], and it was trained on the VGGFace2 dataset [21]. The input shape of the FaceNet is $224 \times 224 \times 3$ (RGB color channel). The encoded face representation vector (output of the FaceNet) is 512-dimensional.

Except for FaceNet, in our framework, other neural networks (E_i, F_i, D_i, and the offset regressor network) are Multilayer Perceptron (MLP) networks. E_i and D_i constitute an VAE network that learns to reconstruct 3D shape S_i. The shape latent regressors F_i are added to replace the E_i to couple the FaceNet encoder with the shape decoders D_i in single-image 3D face reconstruction task (see Figure 3.2). Please see Appendix A.1 for network architectures of E_i, F_i, D_i, and the offset regressor network. $|V_i|$ is the number of vertices of segment i, and d_{z_i} is the dimension of latent representation of segment i. Note that, the output of offset regressor network has 10 dimensions because only five segments need offset along y-axis (up and down) and z-axis (forward and backward).

Assembling Face Segments and Network Fine-Tuning

There are two remaining issues. The first is that we re-centered the S_ks for training the corresponding shape decoders to disentangle the five-parts shapes from the global face shape. Therefore, these decoders are not aware of the relative position of S_ks to S_{rest}. The second issue is that the pre-trained FaceNet is not optimized for 3D face reconstruction.

To address the first issue, we train an offset regressor that takes S_{rest}'s latent representation z'_0 as input to predict an offset $(0, \Delta \mathrm{y_k}, \Delta \mathrm{z_k})$ for each reconstructed segment shape S'_k. Because the human face is bilaterally symmetric, we do not add any offset along the x-axis. After adding the offsets, we can assemble the S_ks with S_{rest} to get a full 3D face $S'_{overall}$. Note that $S'_{overall}$ is likely to have visible segment boundaries without proper blending.

To address the second issue, we fine-tune all the networks except the shape decoders D_is to improve the reconstruction accuracy. Besides using the target shapes as supervision, we leverage the ground-truth 2D face parsing masks as additional supervision. The loss function for fine-tuning is defined as:

$$\mathcal{L} = \lambda_s \mathcal{L}_s + \lambda_p \mathcal{L}_p + \lambda_{KL} \mathcal{L}_{KL}. \tag{3.2}$$

$\mathcal{L}_s$ is the shape loss between $S_{overall}$ and $S'_{overall}$, the computation is the same as $\mathcal{L}_{recon}$. Define $\mathrm{P} \in \mathbb{R}^{\mathrm{H} \times \mathrm{W} \times 5}$ as the ground-truth parsing mask of S_ks, where H and W are the height and width of the mask. P_{ijk} represents the binary label that indicates whether a pixel at image location (i, j) belongs to the k^{th} ($k \in \{1..5\}$) segment or not. We include the estimated expression and pose and use a differentiable renderer to render $S'_{overall}$ to 2D, denoted by P'. The renderer is modified to use the vertex one-hot semantic labels as five-channel colors for shading. To prevent the gradient from vanishing when the displacement between P and P' is too large, we follow the idea in [95] to use Gaussian filter (we set the standard deviation to be 1 and filter size to be 10×10) to soften both P and P'. The parsing loss is then defined as: $\mathcal{L}_p = \|\mathrm{P} - \mathrm{P}'\|_2$. Here, the $\mathcal{L}_{KL}$ acts as a regularization term.

Face Segments Blending

An evident challenge for the segment-based face model is that it is not straightforward to ensure the transition between segments is natural-looking. Simply smoothing the region between different face segments could destroy the high-frequency local structure. Our goal is to preserve the surface details as much as possible so that the final shape will still looks human and to make the transitions between face segments look natural without abrupt boundaries between segments. To achieve this, we use the As-Rigid-As-Possible (ARAP) [170] for mesh deformation optimization in our blending module.

Our blending module will only deform some areas on S'_{rest} that connects it to the five-segments. We call these areas the transition area (see Figure 3.3 right). We define the transition area using the average BFM face shape. The transition area covers the vertices on S'_{rest} whose Euclidean distances are less than $30mm$ from their nearest vertices on the segment boundaries (see Figure 3.3 right, the red lines). The outer contour of the transition area is the outer boundary (see Figure 3.3 right, the blue lines). We use the ARAP method to deform the shape of the transition area. The constraints are formed by two groups of vertices – vertices from the

five-segments lying on the segment boundaries and vertices from S'_{rest} lying on the outer boundary. In our experiments, we run the ARAP optimization for three iterations.

Intuitive Face Editing

We expect the editing to be intuitive. For example, one can increase/decrease the nose width by an adjustable strength. The overall idea is to find mappings between the pre-defined facial feature measurements space and the latent encoding space. Different from [59], our editing is not based on the exact measurement values (e.g., the exact width of the nose bridge). It is based on the direction ($+$ for increase, $-$ for decrease) and strength (how many standard deviations from the mean shape). We refer to some studies on facial feature measurements [46, 91, 147, 150] and derive a list of features for editing purposes (see supplementary material). We grouped the features by the six segments. For example, nose height is a feature of the nose segment.

We denote N as the number of face shapes in the dataset and m as the number of features of the shape S (S can be the shape of any of the six segments). For each S in the dataset, we record the measurements of its features (the measurement is done automatically using pre-defined measurement rules) as a row in the matrix $X \in \mathbb{R}^{N\times m}$, and also record its latent encoding vector z as a row in the latent matrix $Z \in \mathbb{R}^{N\times d_z}$. We compute the average face shape $\bar{S}$ over the dataset and record its latent encoding as $\bar{z}$. Next, we subtract the measurements of average shape $\bar{S}$ from each row in X and denote the resultant matrix as ΔX. Similarly, we subtract $\bar{z}$ from each row in Z and denote the resultant matrix as ΔZ. Assume there is a linear mapping $M \in \mathbb{R}^{m\times d_z}$ such that $\Delta X M = \Delta Z$, we can estimate M by solving the linear system $M = \Delta X^{\dagger}\Delta Z$, where $\Delta X^{\dagger}$ is the Moore–Penrose pseudo inverse of ΔX.

We define the editing controllers as a vector $c \in \mathbb{R}^{1\times m}$. Each value in c controls a corresponding feature. We compute the standard deviation of each column in ΔX, and record it as $\Sigma \in \mathbb{R}^{1\times m}$. Finally, we represent the editing as the modification of latent encoding space: $\tilde{z} = z + \Delta z = z + (c \circ \Sigma)M$, where z is the latent encoding of the original shape S, and $\circ$ is the Hadamard product operator. Using

the shape decoder to decode the new latent encoding $\tilde{z}$, we get the edited shape. We need to compute an M and a Σ for each segment.

Differentiable Blending Module in Shape Adjusting

To further improve the 3D-to-2D alignment accuracy, we propose a novel differentiable version of ARAP in our blending module. Our differentiable ARAP is fully functional, transforming traditional ARAP optimization iterations into a sequence of differentiable operations through which loss function gradients can back-propagate. Our method is different from [246], which has proposed the use of the ARAP energy as a loss function but did not carry out the ARAP optimization itself. In contrast, our method back-propagates through the ARAP algorithm, including estimating neighborhood rotations and solving the linear system. In principle, our ARAP deformer module can be applied to many deep learning-based geometry processing tasks and used as a component in end-to-end training. However, here we use it to fine-tune the editing controllers due to the high vertex count of our facial model.

We define n_p as the number of vertices of the face mesh, and n_c as the number of constraint vertices. We represent the vertex positions of the 3D mesh as a matrix $P \in \mathbb{R}^{n_p \times 3}$, and the constraint vertex positions as a matrix $H \in \mathbb{R}^{n_c \times 3}$. We first build a sparse constraint matrix $C \in \mathbb{R}^{n_c \times n_p}$, such that $C_{ij} = 1$ only if the j^{th} vertex of the mesh is the i^{th} constraint vertex. Then we compute the combinatorial Laplacian of the 3D mesh denoted by $L \in \mathbb{R}^{n_p \times n_p}$. The reason for using the combinatorial Laplacian rather than the cotangent Laplacian is that the former only depends on the mesh topology so that we can reuse it on different 3D faces with the same mesh topology. We define $P' \in \mathbb{R}^{n_p \times 3}$ as the new vertex positions, R as the right-hand-side of linear system $AW = R$, where A is a sparse matrix (3.3) and $(W_{ij})_{i \in \{1..n_p\}, j \in \{1..3\}}$ is P'.

$$A = \begin{bmatrix} L^{\mathrm{T}}L & C^{\mathrm{T}} \\ C & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(n_p+n_c)\times(n_p+n_c)}. \tag{3.3}$$

The estimation of R needs both P and P', thus the optimization of P' is done by alternatively estimating R and solving the linear system $AW = R$ for multiple

iterations. Because A only depends on the mesh topology, the system only needs to be formed and inverted once. Table 3.2 summarizes the mathematical notations used in the proposed differentiable ARAP. Algorithm 1 is the pseudo-code of the proposed differentiable ARAP, where n_{iter} is the number of iterations we would like to run the ARAP optimization. In our experiment, we choose $n_{iter} = 3$. For ease of explanation, we also use a flowchart to visualize the proposed differentiable ARAP algorithm, see Figure 3.4.

Table 3.2: Mathematical notations for differentiable ARAP.

Notation	Space	Description
n_p	$\mathbb{R}$	The number of vertices of the face mesh.
n_c	$\mathbb{R}$	The number of constraint vertices.
P	$\mathbb{R}^{n_p \times 3}$	The vertex positions (coordinates).
H	$\mathbb{R}^{n_c \times 3}$	The constraint vertex positions.
C	$\mathbb{R}^{n_c \times n_p}$	The sparse constraint matrix, such that $C_{ij} = 1$ only if the j^{th} vertex of the mesh is the i^{th} constraint vertex.
L	$\mathbb{R}^{n_p \times n_p}$	The Laplacian matrix of the face mesh.
P'	$\mathbb{R}^{n_p \times 3}$	The new vertex positions of the face mesh.
A	$\mathbb{R}^{(n_p+n_c)\times(n_p+n_c)}$	$A = \begin{bmatrix} L^{\mathrm{T}}L & C^{\mathrm{T}} \\ C & \mathbf{0} \end{bmatrix}$
W	$\mathbb{R}^{(n_p+n_c)\times 3}$	The result matrix of linear system $AW = R$, and the first n_p rows $(W_{ij})_{i\in\{1..n_p\},j\in\{1..3\}}$ is P'.
R	$\mathbb{R}^{(n_p+n_c)\times 3}$	The right-hand-side matrix of linear system $AW = R$.

Note that L is the combinatorial Laplacian ($L = Deg - Adj$, where *Deg* and *Adj* are the degree matrix and the adjacency matrix of the mesh graph topology). Although using cotangent-weight Laplacian (as used in the original ARAP [170]) will give better results, the computational overhead is much higher than using simple combinatorial Laplacian. Therefore, we only use the cotangent-weight Laplacian to derive the final result while using combinatorial Laplacian in fine-tuning. In Algorithm 1, A^{-1} is pre-computed for a given mesh. We treat L and A^{-1} as constant matrices, thus not backpropagating the error through it.

Except for the initial iteration (when $iter$ is 0), we need to estimate the new

Algorithm 1 Differentiable ARAP

Input: P, H, L, and A^{-1}
Output: P'
Require: $n_{iter} \geq 1$
$iter \leftarrow 0$
while $iter \leq n_{iter}$ **do**
  if $iter = 0$ **then**
    $R \leftarrow \begin{bmatrix} L^{\mathrm{T}}LP \\ H \end{bmatrix}$ // Initialize the right-hand-side.
  else
    $R \leftarrow \begin{bmatrix} estimate_rhs() \\ H \end{bmatrix}$ // Estimate new right-hand-side (see Algorithm 2).
  end if
  $W \leftarrow A^{-1}R$ // Solve the linear system $AW = R$.
  $P' \leftarrow (W_{ij})_{i \in \{1..n_p\}, j \in \{1..3\}}$ // Update new vertex positions.
  $iter \leftarrow iter + 1$
end while

right-hand-side R using $estimate_rhs()$. Here, we take advantage of the fact that L is a sparse matrix to reduce computational overhead. We define $N(i)$ as the set of neighbor vertex indices of the i^{th} vertex, and $N(i)_k$ as the k^{th} neighbor vertex index (neighbor vertices arranged in index ascending order) of the i^{th} vertex. The pseudo-code for $estimate_rhs()$ is in Appendix A.2. Note that the pseudo-code only conveys the idea of our algorithm. It is not optimized for a particular programming language. When implementing it, consider using matrix operations to reduce the explicit loops.

The main idea of ARAP is to minimize the energy function (3.4) for each vertex i:

$$\mathcal{E}_i = \sum_{j \in N(i)} w_{ij} ||(p'_i - p'_j) - \mathbf{R}_i(p_i - p_j)||^2, \tag{3.4}$$

where w_{ij} is the Laplace weight of edge ij, p'_i is the new vertex position of i, p_i is the original vertex position of i. The ARAP alternates between updating all the p'_i and $\mathbf{R}_i$ ($\forall i \in n_p$) in each iteration. When p'_i are fixed, $\mathbf{R}_i$ are estimated to look for the best rigid rotation. When $\mathbf{R}_i$ are fixed, the function is minimized by solving the Laplace equation with edge ij rotated.

Unlike our fine-tuning of the network, we do not apply the Gaussian filter on

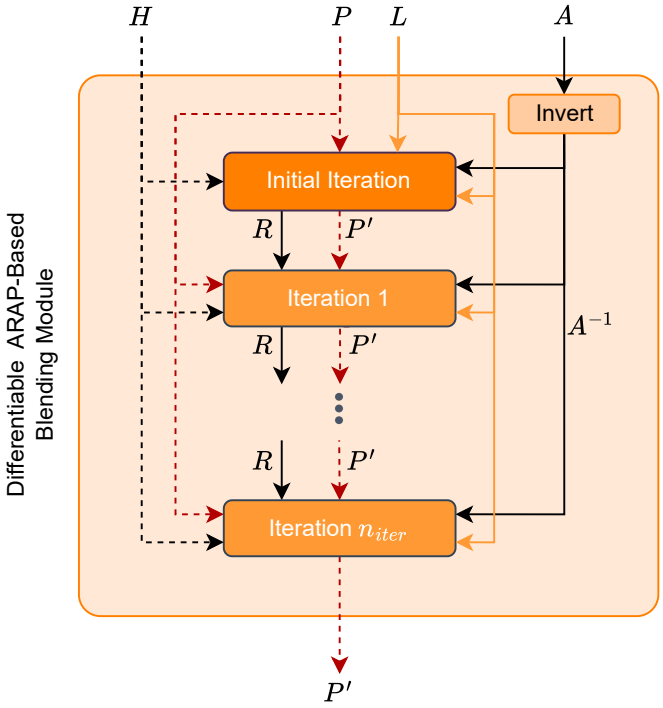


Figure 3.4: Flowchart of our differentiable ARAP-based blending module. Dashed lines indicate that the loss can backpropagate through.

both the ground-truth parsing P and the rendered parsing P', instead we only apply distance-transform to P. We empirically find that this approach leads to a better result. Importantly, we directly adjust the editing controllers instead of adjusting the segment latent representations. In the optimization process, we only minimize $\|\mathrm{P} - \mathrm{P}'\|_2$. The optimization is performed on a single shape each time.

3.3.4 Experiments

Data and Training

The face images we use in our experiments are from CelebAMask-HQ dataset (30,000 images) [102] and FFHQ dataset (70,000 images) [88]. We run Deng et al.'s method (Deep 3DMM) [38] on both datasets to get the estimated BFM coef-

ficients for each image. An alternative option is to use the synthesis-by-analysis method [14] to estimate the coefficients, but it would be slow. Then, we use the BFM coefficients to recover the 3D faces and get 100,000 3D faces. Because the CelebAMask-HQ dataset comes with ground-truth semantic face parsing masks, we use its first 5,000 images to form the validation set. The rest of the 95,000 images and their 3D reconstructions are used to train the models. We use the 25,000 training images from the CelebAMask-HQ dataset and their corresponding semantic parsing masks to fine-tune our network. We also use all the 95,000 3D reconstructions in the training set to compute the mappings for intuitive editing. The loss term weights are set as follows: $\lambda_{recon} = 1$, $\lambda_{KL} = 0.01$, $\lambda_{smooth} = 1$, $\lambda_s = 1$, $\lambda_p = 1$. We use Adam optimizer with a constant learning rate of 10^{-4}. The training batch size is 16. Our framework has an average per-vertex mean squared error of 9.0×10^{-3} on validation data (3D mesh data is measured in millimeters).

3D-to-2D Segments Alignment

In the single-image 3D face reconstruction task, the best that can be done is to make the rendered 3D face look as close as possible to the face in the image. It is challenging for existing global-based 3D face reconstruction methods to make each 3D semantic face segment better align with its corresponding 2D image semantic segment. We presume that the global-based 3D face model has strong constraints to ensure the entire 3D face looks as natural as possible. The drawback is that small face segments such as the eyes and the lips may not look realistic and are prone to be misaligned. In addition, the human visual system is very sensitive to tiny differences in a face. Thus, the global-based methods are usually not able to provide satisfactory reconstructions. Some research works [27, 99] try to mitigate this problem by using more realistic textures to cheat the human viewers. In this work, we take advantage of our segment-based face model to improve the shape accuracy by tuning each 3D semantic segment, so it better aligns with the image. Learning realistic texture will also benefit from better aligned 3D shapes [111]. We conjecture that improved registration of 3D geometry to semantic face segments will assist texture reconstruction models in learning realistic textures, although it is not a focus of the current work.

We use the average Intersection-over-Union (IoU) metric to evaluate the 3D-to-2D alignment accuracy. Table 3.3 compares the deep 3DMM (global method), PB-3DMM, LD-VAE, and the proposed NEO-3DF. Note that both PB-3DMM and LD-VAE are not designed for single-image 3D face reconstruction, and both methods originally use different mesh topologies other than BFM, which we use. To compare with PB-3DMM and LD-VAE, we use the BFM mesh topology and our parsing-based face segmentation. Because both PB-3DMM and LD-VAE are local models, we use the similar network architecture of NEO-3DF to give both methods the single-image 3D face reconstruction ability. For PB-3DMM, the shape decoders are from BFM. For LD-VAE, the shape decoder is pre-trained using the original LD-VAE code, but on our dataset. We fine-tuned both variants under the same setting as training NEO-3DF.

Table 3.3: 3D-to-2D alignment results on validation data. Metric: average IoU.

Part Name	**Deep 3DMM [38]**	**PB-3DMM [59]**	**LD-VAE [49]**	**NEO-3DF**
Eyebrows	0.312	0.302	0.279	**0.363**
Eyes	0.387	0.377	0.372	**0.553**
Nose	0.766	0.728	0.735	**0.789**
Upper Lip	0.483	0.482	0.436	**0.531**
Lower Lip	0.468	0.424	0.418	**0.614**
All Five Parts	0.593	0.566	0.560	**0.616**

For single-sample shape adjusting, we use Adam optimizer with a learning rate of 10^{-3}. We run 20 gradient-descent optimization iterations for each shape, and the shape adjusting takes around 31 seconds for each iteration (on a workstation with one NVIDIA Tesla V100-PCIe GPU). On the validation data, the average IoU of five-segment shapes after fitting is 0.683 (discussed in Ablation Study), which is 11% higher than the raw network prediction (0.616 as shown in Table 3.3). Figure 3.5 shows some visualization of the alignment performance comparison. For each method, we use one column to show the rendered 3D face on top of the original image, and another column to show the union minus the intersection of P and P'. If the shapes of the face segments are better aligned to the 2D image, the black areas should be less visible.

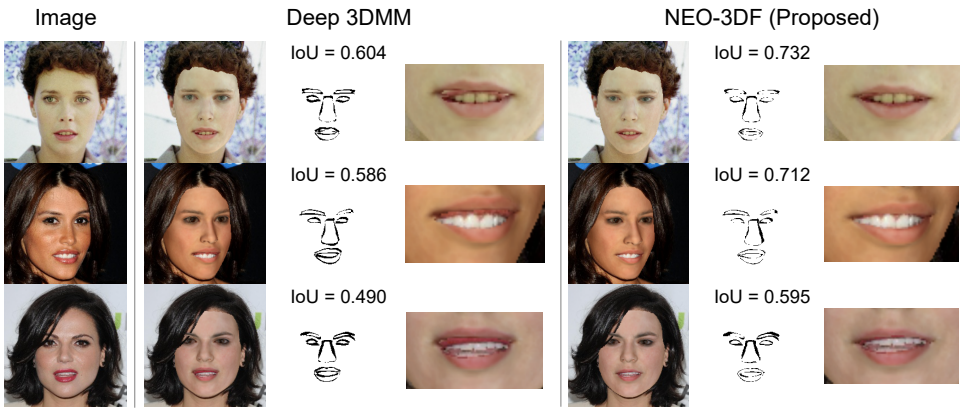


Figure 3.5: 3D-to-2D semantic face segments alignment results. Compared with Deep 3DMM, we can see that our method gives better aligned facial segments (yields higher segmentation accuracy, as measured by IoU) and produces more consistent rendered results with original images.

Intuitive 3D Face Editing

We compare our method with global VAE and LD-VAE [49]. We use the same MLP network structure for both global VAE and the LD-VAE, where the decoder structure is the same as non-linear 3DMM's shape decoder [186]. Our shape decoders have a similar decoder structure (same number of layers and hidden neurons), but each has a smaller output layer because the shape of a segment has fewer vertices than the entire face shape. Please refer to Table A.5 in Appendix A.3 for the list of selected facial features used for editing.

We group the controllers by segments. In each group, we tune each controller separately to be -3σ and $+3\sigma$, then aggregate all the edited shapes and show the maximum change of each vertex location (Euclidean distance measured in millimeters) from the original mean face shape. The results are visualized as rendered 2D heatmaps (see Figure 3.6). In addition, we show the 2D heatmaps of each individual controller in Appendix A.3 (Figure A.1).

Figure 3.7 shows the results of applying the same set of edits on the nose (*Tip* -3σ, *Breadth* $+3\sigma$, *Bridge Width* -3σ) using different methods. To demonstrate the application of our framework in customizing a reconstructed 3D face, we ran-

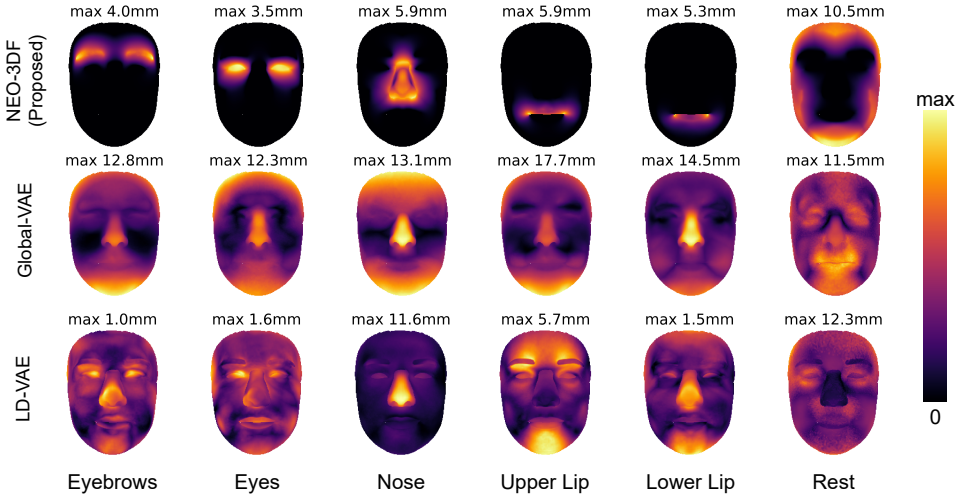


Figure 3.6: Vertex location change heatmaps. For visualization purposes, the values in each heatmap are normalized to be from 0 to 1.0. The value above each heatmap indicates the highest value of that heatmap. We can see that our method gives near-perfect disentanglement while the other methods cannot isolate local edits well.

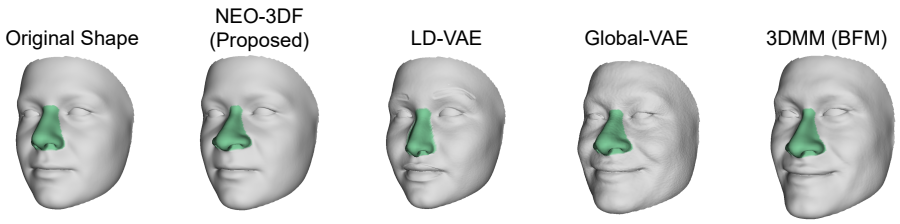


Figure 3.7: Nose shape editing comparison.

domly apply a sequence of edits to the reconstructed faces. Figure 3.8 shows some example intuitive editing results.

Ablation Study

We first investigate the necessity of employing the proposed offset regressor and blending module. Figure 3.9 shows the original shape and the editing we made (lift the nose tip) using our unmodified framework, our framework without offset regressor, and our framework without blending module. Note that, in this exper-

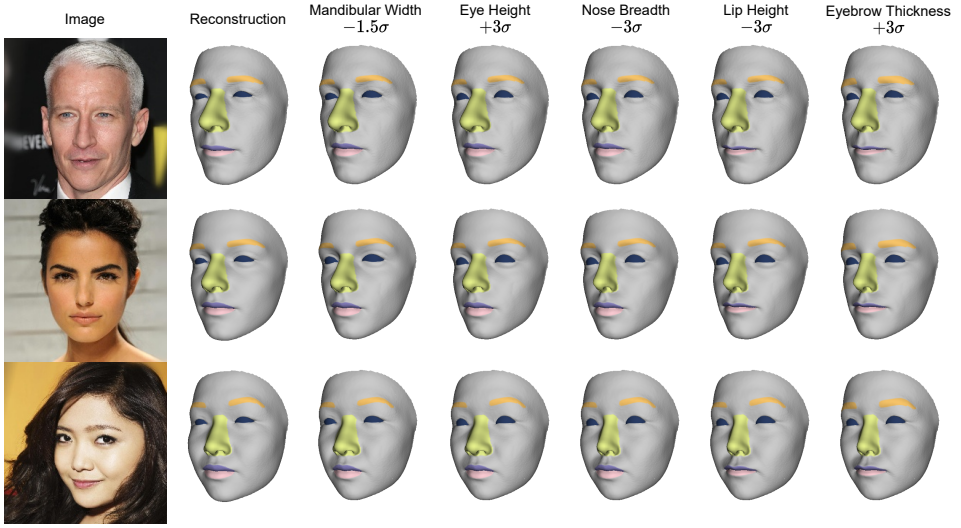


Figure 3.8: Reconstruction and further editing examples. The edits are made cumulatively from left to right. Editing via our framework is more intuitive and expressive since subsequent changes (on different segments) will not affect earlier ones.

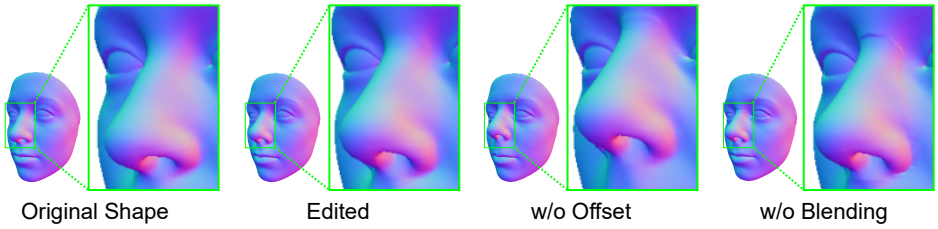


Figure 3.9: Ablation study shows that the offset regressor and blending module are necessary for our framework. Without the offsets, alignment between segments is compromised, while visible seams appear around the edited segment without the blending module.

iment, only the nose is editable. Without the offset regressor, we need to use the absolute vertex locations to train the five-segments decoders. So the relative location of a five-part shape is still entangled into the latent representation. As a result, after removing the offset regressor, we see that the entire nose moves up when lifting the nose tip. The issue of eliminating the blending module is evident, as we can see the juncture around the nose.

Table 3.4: Single-sample automatic 3D part shapes adjustment results. (Metric: IoU averaged on validation data)

	w/o Blending	**w/ diff. ARAP Blending**
Latent Representations Only	0.621	0.626
Editing Controllers Only	0.581	0.566
Latent Representations & Offsets	0.629	0.632
Editing Controllers & Offsets	0.649	**0.683**

We also investigate different ways of improving the 3D-to-2D alignment in our fitting-based single shape adjusting (see Table 3.4). We find that with the help of our differentiable ARAP, the best results are obtained by the fine-tuning of the editing controllers and the shape offsets. This yields 7% higher IoU than by directly adjusting the latent encodings and offsets (IoU is 0.632). Our explanation for this finding is that the editing controllers' space is less complex and more interpretable than the latent encoding space, thus reducing the learning complexity. Interestingly, we noticed that it does not work if we do not optimize the offsets and only optimize the editing controllers. We presume that the face segments must be roughly aligned to provide useful loss information.

3.3.5 Summary

The proposed 3D face creation framework NEO-3DF models the different parts of the 3D face independently. NEO-3DF is the first method that couples single-image 3D face reconstruction and 3D face intuitive editing. To train NEO-3DF, we created a 3D face dataset using existing large 2D face image datasets and an off-the-shelf 3D face reconstruction method named deep 3DMM. The face segment-based model of the 3D face structure makes face editing more intuitive and user-friendly. It also allows fine-tuning the reconstructed 3D face by using 2D semantic face segments as guidance and yields 14% improvement in IoU. We also proposed a differentiable version of ARAP to obtain an end-to-end trainable framework, which enables the automatic adjusting of the variables in the editing controllers. This resulted in an additional 5% improvement in IoU. The main limitation of our current framework is that the differentiable ARAP is computationally expensive

and only used to fine-tune the result at the last construction stage. We will explore a less expensive method for blending the face segments in the future. We believe our concept of combining the 3D face editing and reconstruction also sets a new direction in 3D face reconstruction research.

3.4 Method 2: Weakly Supervised Neural 3D Face Shape Stylization

3.4.1 Overview

3D Face shape stylization refers to transforming a realistic 3D face shape into a different style, such as a cartoon face style. To solve this task, this thesis proposes modeling it as a deformation transfer problem. This approach significantly reduces labor costs, as the artists would only need to create a single template for each face style. Realistic facial features of the original 3D face, e.g., the nose or chin shape, would thus be automatically transferred to those in the style template. Deformation transfer methods, however, have two drawbacks. They are slow and they require re-optimization for every new input face. To address these weaknesses, we propose a neural network-based 3D face shape stylization method. This method is trained through weakly supervised learning, and its template's structure is preserved using our novel template-guided mesh smoothing regularization. Our method is the first learning-based deformation transfer method for 3D face shape stylization. Its employment offers the useful and practical benefit of not requiring paired training data. The experiments show that the quality of the stylized faces obtained by our method is comparable to that of the traditional deformation transfer method, achieving an average Chamfer Distance of approximately $0.01mm$. However, our approach significantly boosts the processing speed, achieving a rate approximately 3,000 times faster than the traditional deformation transfer.

The main contributions of this work are summarized as:

1. We propose the first learning-based method for the 3D face stylization task that requires only one artist-designed target template face per style. The proposed method is based on weakly-supervised learning to address the lack of paired source and artist-designed faces as training data;

2. We propose a novel template-guided mesh smoothing filter loss. We demonstrate that it is more robust in preserving the mesh structure than the traditional Laplacian smoothing when a template mesh is available. Experiment also shows that the proposed loss is highly efficient;

3. The proposed method achieves results comparable to the traditional deformation transfer but is around 3,000 times faster. We demonstrate that real-time processing enables seamless user interaction through a face editing application.

3.4.2 Background

For clarity, we consider the face features in the source shape as deformations to the average source face. Existing works on transferring deformations from the source to the target face can be broadly categorized into two groups: traditional deformation transfer methods [134, 171] and learning-based methods [10, 67, 197, 229, 240].

Traditional Deformation Transfer Methods

The first traditional deformation transfer method (DT) for triangle meshes was introduced by Sumner et al. [171]. It is notable for its ability to handle different mesh topologies between the source and target, thus facilitating the reuse of manually designed deformations like blendshapes. The core approach of DT involves establishing triangle-level correspondences between meshes, allowing for the transfer of deformations characterized by triangle translations and rotations. An optimization process then ensures local structure consistency in the resultant mesh. Building on this, Onizuka et al. developed the landmark-guided DT (LDT) to enhance face feature transfer in human-to-avatar deformations, by incorporating user-defined face landmarks into the DT optimization process [134]. However, the DT's optimization process is computationally expensive and cannot be used in real time, which poses a significant challenge given that 3D avatar applications typically serve a large number of users, making efficiency in 3D face stylization crucial. The cage-based deformation method (CBD) [151] is a less computationally demanding alternative to DT. However, CBD is mainly for transferring relatively large-scale deformations

such as face expressions rather than subtle deformations that are needed in 3D face stylization.

Learning-Based Deformation Transfer Methods

The learning-based DT methods can significantly reduce the processing time. Early work in learning-based 3D face stylization approached this problem as a supervised learning task and required a set of reconstructed 3D faces and a corresponding set of artist-designed style faces for training [240]. Thus the creation of paired training data is unfortunately costly. Basset et al. introduced a neural deformation transfer method via an encoder-decoder network [10]. Their method requires the source and target meshes to have an identical topology. This is not easy for the 3D face stylization task because the artist-designed faces usually have different topologies. Some face expression transfer methods regard the source-to-target shape deformation transfer as the domain transfer in learned latent spaces [197, 229]. These methods first train the shape auto-encoders for both source faces and target faces respectively, then learn a mapping between the latent spaces of both auto-encoders. Training a target auto-encoder typically requires a diverse dataset of stylized faces featuring a range of local shape deformations. However, in our specific face stylization problem, there is a lack of stylized faces of varying face features for training a target auto-encoder in a supervised manner. It is also worth noting that Gao et al. proposed a deformation transfer method using a cycle-consistent generative adversarial network (GAN) to learn the mapping between source and target shape spaces without requiring paired source-target data [55]. However, this method still relies on having a set of target shapes with varying deformations, which is not available in our task. Han et al.'s exemplar-based 3D portrait stylization utilizes 2D face landmarks (detected from the 2D artistic face image) to guide the deforming of the reconstructed 3D face [67]. Since the 2D landmarks lack 3D information, this often results in a lack of 3D consistency in the stylized 3D faces.

3.4.3 The Proposed Method

Problem Formulation

We define the 3D triangle mesh as $m = (V, T)$, where $V \in \mathbb{R}^{N\times 3}$ and $T \in \mathbb{R}^{M\times 3}$ are the 3D vertex locations and the triangles represented by vertex indices, N and M represent the number of vertices and triangles respectively. Assume meshes m' and m have identical topology (we can deform m to m' without changing the mesh connectivity). For convenience, we define Δm as the vertex offsets that can deform m to m':

$$m' = m + \Delta m. \tag{3.5}$$

Note that, in this deformation process led by vertex offsets, only the vertex locations are changed while the mesh topology defined by triangles remains the same.

The deformation transfer method considers two types of meshes: source and target respectively [171]. In the 3D face stylization task, we define a reconstructed human face mesh as the source mesh m'_s, and its stylized mesh as the target mesh m'_t. Same as in the case of deformation transfer, we require to have a **template source mesh** m_s and a **template target (style) mesh** m_t. The mesh topologies of m_s and m_t do not need to be the same. We can observe the offsets Δm_s between m'_s and m_s. The goal is to find the corresponding Δm_t that can transform m_t into m'_t, where the style defined by m_t should be preserved and the face identity features defined by m'_s should be transferred onto m'_t. Figure 3.10 demonstrates the translation of vertex offsets. In this example, we transfer a kid's face to a cartoon-styled face template.

In our method, we use the 3D morphable face model (3DMM) to represent the source faces. The 3DMM is a statistical model of the 3D human faces, which can be manipulated to generate a wide range of realistic 3D faces by varying a set of parameters [14]. We use 3DMM mainly because it supports 3D face reconstruction and the reconstructed faces have the same mesh topology. We use the average face mesh of 3DMM as the template source mesh m_s. The target style template mesh m_t is handcrafted by the 3D artist.

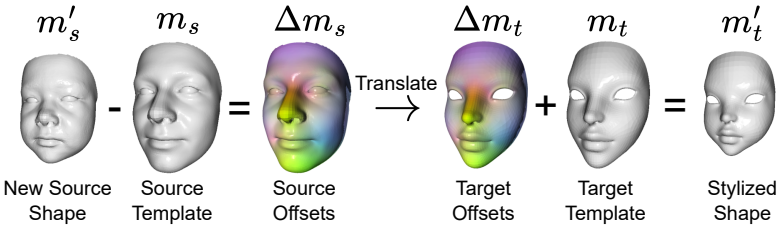


Figure 3.10: Demonstration of the vertex offsets translation. The vertex offsets are represented by RGB colors on the template mesh. Note that, the topologies of source and target meshes do not need to be the same.

Pseudo-Label Generation

Following the traditional deformation transfer methods [134, 171], we first find the correspondence between the template human face mesh m_s and the style template mesh m_t. The m_t is first scaled and aligned to match m_s. Then we solve the minimization problem as defined in the traditional deformation transfer to deform the source template mesh to the target template mesh [171]. Different from the traditional deformation transfer, we do not find the triangle correspondences, rather we find the vertices correspondences. The vertex correspondences are derived by finding the closest vertex on m_s to each of the vertex on m_t. Therefore, we can represent the vertex correspondences as a many-to-one mapping between m_s and m_t, where each vertex in V_t has exactly one correspondence in V_s. Then we can define the mapping as a function that transforms any observed deformation on m_s represented by the offsets Δm_s into Δm_t:

$$\mathcal{M}(\Delta m_s) = \Delta m_t. \tag{3.6}$$

Note that we only need to compute the vertex correspondences once for each different style template. We name the offsets Δm_t derived through Equation 3.6 the **naive offsets**, as they often introduce undesired artifacts in the resulting stylized face. However, we use Δm_t as the pseudo-label in our weakly-supervised learning to reduce the cost of training.

We use vertex correspondences rather than triangle correspondences because

the latter comes with the issue that multiple triangles could share the same vertex and thus requires complex optimization which is the essence of the traditional deformation transfer method [171]. However, the offsets Δm_t derived by Equation 3.6 are not optimal and usually cause observable artifacts in the deformed template mesh m'_t. Therefore we propose to train a neural network model that aims to improve the quality of the offsets.

Neural Network Architecture

Figure 3.11 shows our proposed neural network-based offsets transfer framework. We first train an auto-encoder that reconstructs the source offsets:

$$D_s(E_s(\Delta m_s)) = D_s(z_s) = \Delta\hat{m}_s, \tag{3.7}$$

where E_s and D_s are the encoder and decoder respectively, $z_s \in \mathcal{Z}$ is the latent encoding of Δm_s, and $\Delta\hat{m}_s$ is the reconstructed offsets. The purpose of training E_s and D_s is to learn a latent representation space $\mathcal{Z}$, that can be reused by our subsequent style translation decoder D_t. We name $\mathcal{Z}$ as the shared latent space, as it is shared by different decoders (D_t) for different styles to keep the deformation visually consistent. We first train E_s and D_s, then we reuse the trained E_s in an encoder-decoder to train D_t. During the training of D_t, we do not retrain E_s. The purpose of D_t is to translate z_s into $\Delta\hat{m}_t$, the desired target offsets.

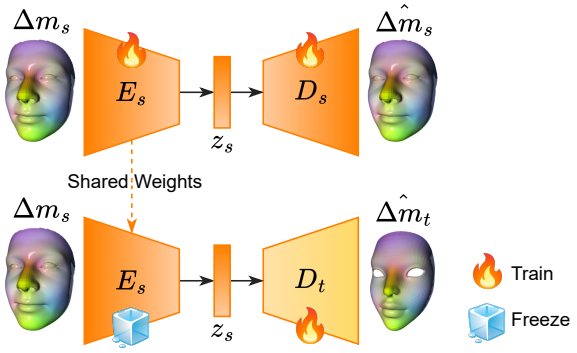


Figure 3.11: Our neural network-based offsets translation framework for 3D face stylization.

3.4.4 Curvature Preservation

To realize our goal of improving the quality of the translated target offsets Δm_t, we consider preserving the local rigidity and curvature information while translating Δm_s to $\hat{\Delta m_t}$. Therefore, we propose to borrow the ideas from two powerful mesh optimization methods: the As-Rigid-As-Possible (ARAP) [170] and the Laplacian mesh optimization [132]. Huang et al. [75] proposed the ARAP regularization loss based on the ARAP's optimization objective function. It can help the decoder preserve the generated shape's local rigidity while being consistent with other loss terms during training. The ARAP regularization focuses on maintaining the rigidity of the mesh, it does not incorporate a smoothing mechanism to reduce the noise. Therefore, we propose an additional curvature-preserving regularization loss that can further reduce the noises and preserve the quality of the mesh structure. This novel approach incorporates our newly developed mesh smoothing filter, which leverages curvature information as a smoothing criterion, inspired by Laplacian mesh optimization [132]. This filter facilitates the computation of a smooth loss that not only preserves the curvature of the style template mesh but also effectively eliminates artifacts, such as spikes, caused by Equation 3.6.

Template-Guided Mesh Smoothing Filter Loss

To prepare for the formulation of our smoothing loss function, we first review the traditional cotangent-weight Laplacian smoothing filter, which is known for preserving local structures [119, 132]. For ease of explanation and subsequent use, we present a slightly adapted form. The Laplacian matrix of mesh m is a squared matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$:

$$\mathbf{L}_{ij} = \begin{cases} w_{ij} & \text{when } v_i \text{ and } v_j \text{ has an edge;} \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

where w_{ij} is the weight of an edge ij. For cotangent-weight Laplacian:

$$w_{ij} = cot(\alpha_{ij}) + cot(\beta_{ij}), \tag{3.9}$$

where α_{ij} and β_{ij} are the angles relative to edge ij. See Figure 3.12 for an example.

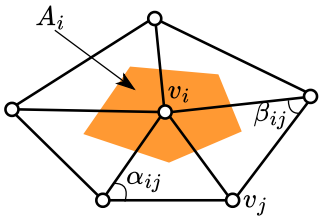


Figure 3.12: Demonstration of a small part of a mesh where the triangles share the same vertex v_i. The orange area A_i is the Voronoi area around vertex v_i.

Define $\mathcal{N}_i$ as the set of one-ring neighbor vertex indices of v_i, and A_i as the Voronoi area [5] around v_i. Then the approximation of the mean curvature normal at v_i is:

$$\overrightarrow{p_i} = \frac{1}{4A_i} \sum_{j \in \mathcal{N}_i} \mathbf{L}_{ij}(v_j - v_i). \tag{3.10}$$

According to Equation 3.10, smoothing a mesh based on the cotangent-weight Laplacian with curvature information can be seen as optimizing the vertex locations by minimizing the following loss:

$$\mathcal{L}_{\text{lap}} = \frac{1}{N} \sum_{i=1}^{N} \| \frac{1}{4A_i} \sum_{j=1}^{N} \mathbf{L}_{ij}(v_j - v_i) \|_2. \tag{3.11}$$

However, in our case, the translated offsets Δm_t contain noisy outliers that "poison" the computed Laplacian and curvature information. If we simply use Equation 3.11 to smooth the deformed mesh $m_t + \Delta m_t$, the result tends to have strong artifacts (see the left mesh in Figure 3.13a). Moreover, when the mesh has fewer vertices, the mesh boundaries tend to get destroyed (see the left mesh in Figure 3.13b). To address these issues, we propose an improved version of the Laplacian smoothing loss. Our approach makes two distinctions. First, instead of computing the Laplacian for the current mesh to be smoothed, we use the Laplacian computed on the template mesh. Second, we prevent smoothing the vertices on the

mesh boundaries by modifying the Laplacian matrix:

$$\mathbf{L}_{ij} = \begin{cases} 0 & \forall i \in boundary, j \in \{1, ..., N\} \setminus \{i\}; \\ 1 & \forall i \in boundary, j = i, \end{cases} \tag{3.12}$$

where $boundary$ is the set of boundary vertex indices. It's important to note that we do not need to manually find the boundary vertices. Instead, these vertices can be automatically detected based on the characteristic that an edge connecting two boundary vertices is exclusive to a single triangle facet. The smoothing loss is still computed through Equation 3.11. The right meshes of Figure 3.13 show that the mesh structures can be well preserved using the proposed template-guided smoothing filter.

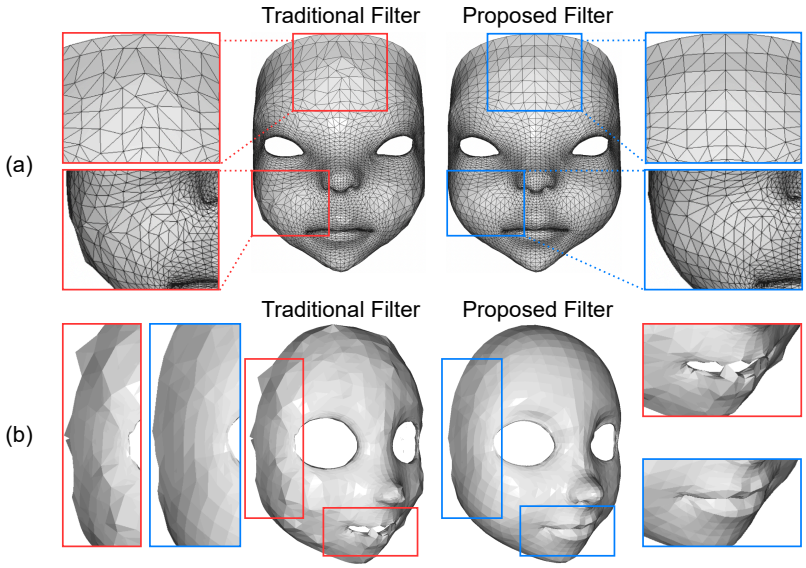


Figure 3.13: Comparisons of the deformed mesh smoothed by the traditional Laplacian smoothing filter and the proposed smoothing filter.

Overall Loss Function

The purpose of auto-encoder $D_s(E_s(\cdot))$ is to learn a source offsets representation in the latent space. The purpose of $D_t(E_s(\cdot))$ is to learn to translate the source offsets into target offsets. Assume Δm represents the ground truth offsets, $\hat{\Delta m}$

represents the network predicted offsets. The offset loss is defined as:

$$\mathcal{L}_{\text{offset}} = \|\Delta m - \hat{\Delta m}\|. \tag{3.13}$$

The reason for choosing the L1 norm is that the L1 norm is less sensitive to the noise in Δm_t. When training the source auto-encoder $D_s(E_s(\cdot))$, $\mathcal{L}_{\text{offset}}$ servers as a supervised loss, while when training the target decoder D_t, $\mathcal{L}_{\text{offset}}$ servers as a weakly supervised loss.

Besides the smoothing regularization loss $\mathcal{L}_{\text{lap}}$ we propose in Section 3.4.4, we use the ARAP regularization loss as in [75] to enforce the local rigidity of the results. We simplify the definition of the ARAP regularization loss as:

$$\mathcal{L}_{\text{arap}} = \|D(z + \delta z) - 2D(z) + D(z - \delta z)\|^2 + r_R. \tag{3.14}$$

The first term is to promote the smoothness of the decoder network concerning the latent space, where $z \sim p(z|m'_s; E_s)$ and δz is a small value. The second term r_R is the local-rigidity preservation loss based on ARAP's energy but only depends on the Jacobian of the decoder.

The final loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{offset}} + \lambda_{\text{lap}}\mathcal{L}_{\text{lap}} + \lambda_{\text{arap}}\mathcal{L}_{\text{arap}}. \tag{3.15}$$

3.4.5 Experiments

Data

We use five mesh topologies in our experiments. The real human face (source) mesh's topology is from the Basel Face Model (BFM) [139], an open-source implementation of 3DMM built on 200 scanned 3D faces. We prepared four artist-designed style template meshes. The mesh topology specifications are in Table 3.5.

To collect the deformed source shapes of varying identities, we run an off-the-shelf 3D face reconstruction method [38] on the FFHQ dataset [87] to get 70,000 reconstructed 3D face shapes. We use the first 60,000 shapes for training, the other 5,000 for validation, and the rest for evaluation.

Table 3.5: Mesh topology specifications.

Mesh	Vertices	Triangles
BFM (Source)	35,709	70,789
Cartoon Adult Face (Target Style 1)	3,100	5,992
Caricature Face (Target Style 2)	3,007	5,816
Cartoon Kid Face (Target Style 3)	1,679	3,194
Cartoon Minimalism Face (Target Style 4)	3,136	6,070

Implementation Details

The networks are multi-layer perceptron (MLP) networks. Both the encoder and the decoder have two hidden layers each. Each hidden layer has 512 neurons. Thus the latent code z_s is 512-dimensional. In our experiments, we set $\delta z = 10^{-3}$, $\lambda_{\text{arap}} = 0.05$ and $\lambda_{\text{lap}} = 10^{-3}$. The training batch size is set to be 32. We train the source auto-encoder and the style decoders for 50 epochs each. We use Adam optimizer with an initial learning rate of 10^{-4} and gradually decrease to 10^{-5}. The training is performed on one NVIDIA RTX A6000 (48GB) GPU. The total time for training the source auto-encoder is around 27 hours, and around 15 hours, 14 hours, 9 hours, and 15 hours for training the style decoders for four styles respectively. The CPU we use in our experiments is an AMD Ryzen Threadripper PRO 5995WX (3.5 GHz, 64-Cores).

Time Comparison

We compare the running time of stylizing a single 3D face using landmark-guided deformation transfer (LDT) [134] and our proposed neural deformation transfer. Table 3.6 displays the time used in the LDT optimization process and our neural network's inference process. We run both methods on the CPU for a fair comparison. The results are averaged on 100 rounds. We observe a significant reduction in stylization time consumption using the proposed method, which is around 3,000 times faster than the traditional method.

We also compare the time consumption used in computing the ARAP regularization loss $\mathcal{L}_{\text{arap}}$ and the proposed regularization loss $\mathcal{L}_{\text{lap}}$. The computations are conducted on the same GPU as we use in training. We set the batch size to 32, with $\mathcal{L}_{\text{arap}}$ averaging approximately 212.62 milliseconds per batch and $\mathcal{L}_{\text{lap}}$ requir-

Table 3.6: Stylization time comparisons.

Style	LDT [134]	Ours (Inference Time)
Style 1	20.537s	**7.02ms**
Style 2	21.137s	**7.21ms**
Style 3	20.109s	**6.53ms**
Style 4	23.942s	**7.39ms**

ing only about 17.88 milliseconds. Fig 3.14 shows the times used for different loss term computations on different style templates.

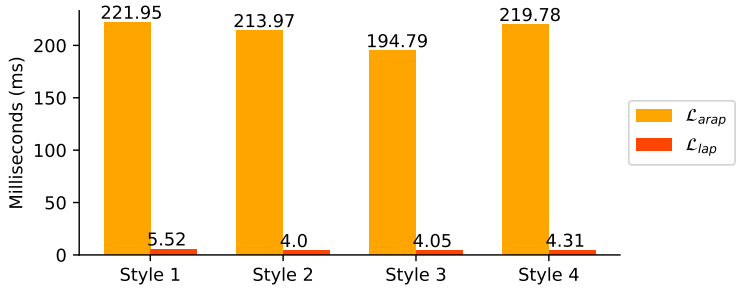


Figure 3.14: Loss term computation time comparisons (in milliseconds).

Stylization Results

In our experiment, we regard the LDT as the upper bound and the naive offset transfer as the baseline. We aim to improve the naive offsets to get a closer result to the LDT. Note that, we do not need to use traditional deformation transfer to generate training targets, which is time-consuming. Instead, using the naive offsets as pseudo-targets and additional local-structure preserving regularization, we can achieve similar results to traditional deformation transfer. Figure 3.15 shows an example of a kid's face stylized by different methods. As shown, the naive vertex offset transfer introduces an unsmooth surface to the result. In contrast, the proposed neural style transfer method derives a much closer result to the LDT.

To prove that facial identity features can be consistently transferred to different styles, we design an experiment that first uses Principal Component Analysis (PCA) to derive the principal components of the latent codes z_s on our test set.

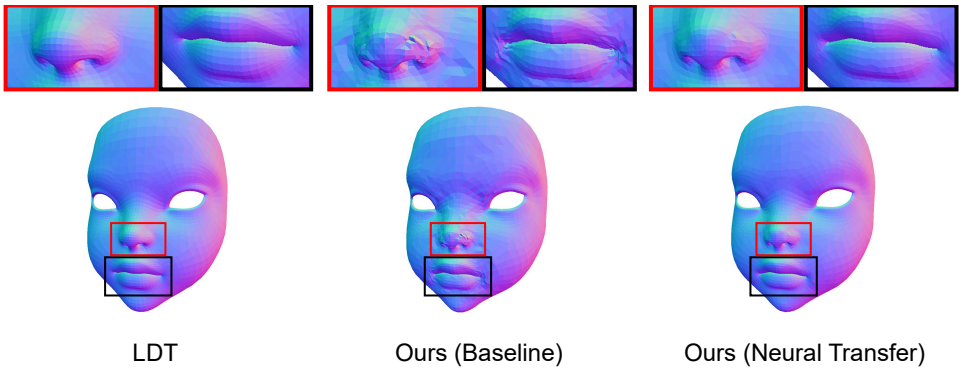


Figure 3.15: Comparison of landmark-guided deformation transfer (LDT) [134], our naive offset transfer baseline, and our proposed neural transfer method.

Then we use each individual principal component to synthesize new latent code by $z'_s = \bar{z}_s + \gamma \text{PC}_i$, where $\bar{z}_s$ is the mean latent code, PC_i is the i^{th} principal component and γ is the factor used to highlight the effect of PC_i. Finally, we use the trained decoder to get the corresponding decoded shape of z'_s. The decoded shape is a deformed version of the mean shape (decoded using $\bar{z}_s$). We set $\gamma = 10$ to make the deformation more visible. Figure 3.16 shows the effects of the first six principal components. We also visualize the shape deformation via color. As shown, the shape deformations on our stylized faces are consistent with the deformation on the source face, meaning that the facial identity features are effectively and consistently transferred across different styles, demonstrating the robustness of our method in preserving identity characteristics while applying stylistic transformations.

Figure 3.17 showcases some stylization results on reconstructed faces (source) from the proposed neural style transfer method. Please visit our project homepage for more results.

Ablation Study

We investigate the impact of our regularization loss terms on the overall performance of the model.

We first demonstrate the advantage of using the proposed regularization loss,

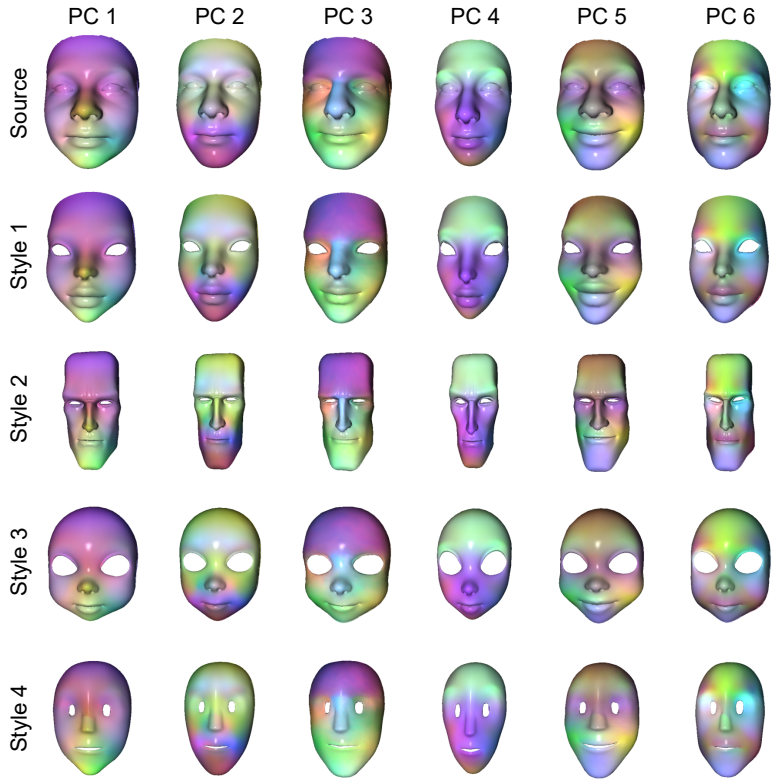


Figure 3.16: The effects of the first six principal components (PCs) on the decoded shapes. To better show the deformation, we visualize the 3D XYZ offsets from the mean shape via RGB color. As shown, in each column, the deformations across different face styles are consistent concerning the facial regions.

$\mathcal{L}_{\text{lap}}$, through the results depicted in Figure 3.18. We use the result of the LDT method as a reference. Additionally, we present outcomes from two variants of our method: one without using the proposed regularization loss (w/o $\mathcal{L}_{\text{lap}}$), and our complete method (w/ $\mathcal{L}_{\text{lap}}$). As we can see, without using the proposed $\mathcal{L}_{\text{lap}}$ leads to noticeable distortions in the mesh structure, particularly in regions such as the forehead and chin in this example. In contrast, both the LDT and our complete method succeeded in keeping the structure smooth and consistent with the template. This experimental result further suggests that the proposed smoothing

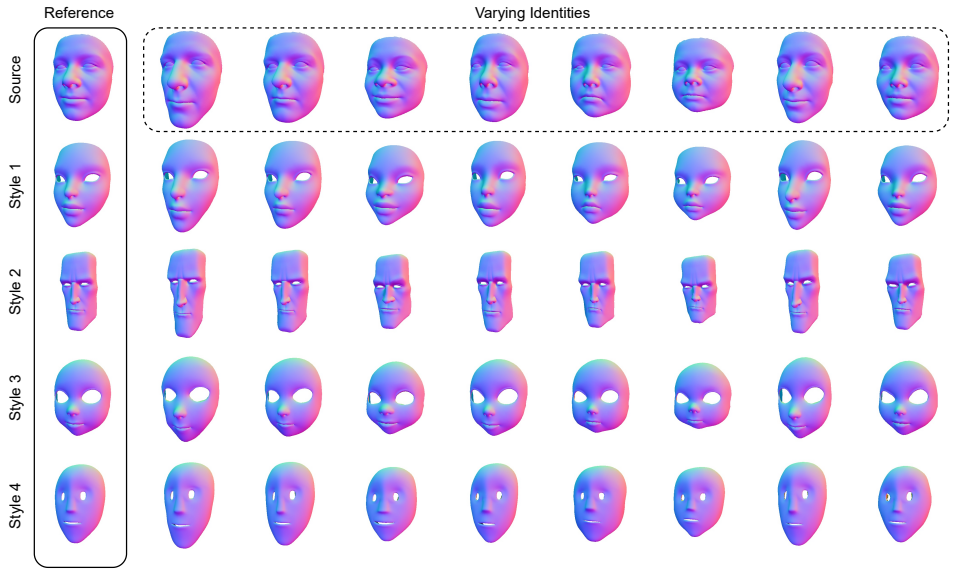


Figure 3.17: Stylization results of reconstructed faces. The first row shows the source faces, where the reference source face is the BFM's mean face and the rest are the reconstructed 3D faces with varying identities. The last four rows show the corresponding stylized faces.

regularization loss can preserve the original mesh structure.

Then, we conducted a quantitative evaluation of the regularization terms using several metrics: mean Chamfer Distance per vertex (CD) [4], mean Euclidean Distance per vertex (ED), and mean Maximum Distance (MD). Both CD and ED are averaged per vertex and, therefore are less sensitive to outliers (such as spikes). However, the outliers can make the most impact on visual perception. To better capture the outliers, we introduce the MD metric, which measures the maximum distance between corresponding vertices on the target and predicted meshes. Here, the targets are results from LDT. Table 3.7 presents the results. We find that in style 1 and style 2, using both regularization loss terms leads to more improvement than in style 3 and style 4. This is because of the inherent simplicity and smoother surfaces of styles 3 and 4, which make them less receptive to the benefits of our regularization strategies. Despite this, the combined use of $\mathcal{L}_{\text{arap}}$ and $\mathcal{L}_{\text{lap}}$ consistently results in the best performance across all styles. However, given the higher

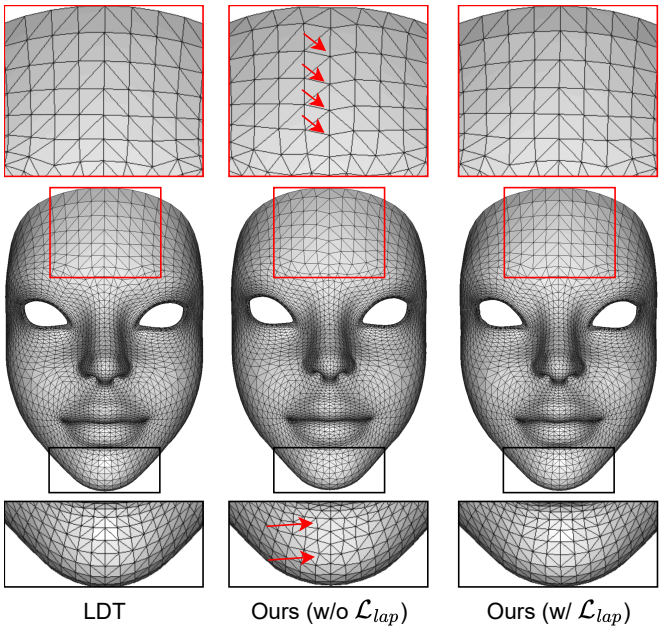


Figure 3.18: Qualitative results of ablation study on regularization loss terms. Arrows highlight the inconsistent structures.

computational cost of ARAP loss, it is reasonable to exclude this term in scenarios where its benefits are marginal. This trade-off allows for significant savings in computational time while maintaining acceptable performance.

Table 3.7: Ablation study on regularization loss terms (measured in millimeters).

Loss Terms			*Style 1*			*Style 2*			*Style 3*			*Style 4*		
$\mathcal{L}_{\text{offset}}$	$\mathcal{L}_{\text{arap}}$	$\mathcal{L}_{\text{lap}}$	**CD** ↓	**ED** ↓	**MD** ↓	**CD** ↓	**ED** ↓	**MD** ↓	**CD** ↓	**ED** ↓	**MD** ↓	**CD** ↓	**ED** ↓	**MD** ↓
✓	✓	✗	0.0235	1.04	5.88	0.178	5.91	9.11	0.0747	2.21	4.06	0.111	**2.74**	4.66
✓	✗	✓	0.0177	0.963	2.51	**0.175**	**5.87**	7.50	0.0749	**2.20**	4.13	0.111	2.75	4.79
✓	✓	✓	**0.0176**	**0.958**	**2.41**	**0.175**	**5.87**	**7.38**	**0.0746**	**2.20**	**3.98**	**0.110**	**2.74**	**4.46**

Application: Stylized Face Editing and Expression Control

Our face stylization approach seamlessly integrates with the existing 3D face editing method, NEO-3DF [216], enabling intuitive modifications to stylized faces. The process involves using NEO-3DF's editing functions to alter the BFM face

shape features, followed by applying our face stylization network to stylize the edited BFM face. Figure 3.19 showcases the results after applying editing on some face features including facial width, nose width and height, and thickness of lips. This demonstration suggests that the proposed neural 3D face stylization method enables reusing the existing 3D face editing system for different 3D face templates. Importantly, as our neural network inference time is only around several milliseconds (see Table 3.6), the editing can be transferred in real time. We also provide an example demonstrating that our stylized 3D faces are compatible with traditional expression blendshapes for animation. Using deformation transfer, we adapt a set of artist-created expression blendshapes for each style and apply them to our stylized faces. Figure 3.20 presents sample results, with additional examples available on our project homepage.

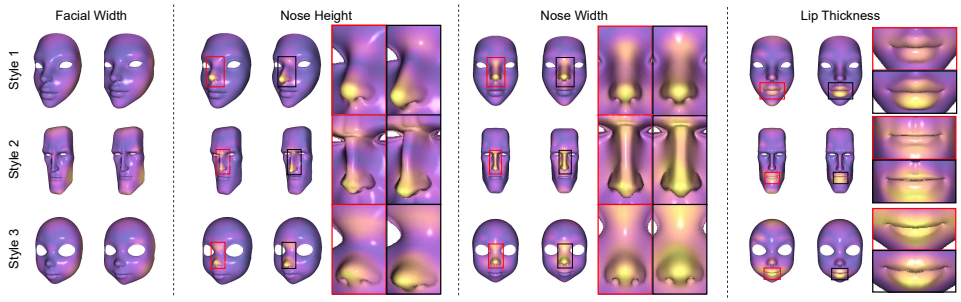


Figure 3.19: Examples of editing the local face features on the stylized faces. We first use NEO-3DF [216] to edit the local face features of the corresponding BFM face, then use our neural style transfer method to stylize the edited BFM face. Colors represent how much the changes are from the template face, with purple indicating minimal alterations and yellow denoting significant modifications.

Reconstruct the Original Face from the Stylized Face

Robustness of the shared latent space can be demonstrated by the bidirectional mapping. In this experiment, we reversed the task by training an encoder-decoder for each style to reconstruct the original faces in the source domain (BFM mesh) from the stylized faces, demonstrating that the stylized faces preserve identity information. The encoder-decoders are trained using the same settings as before,

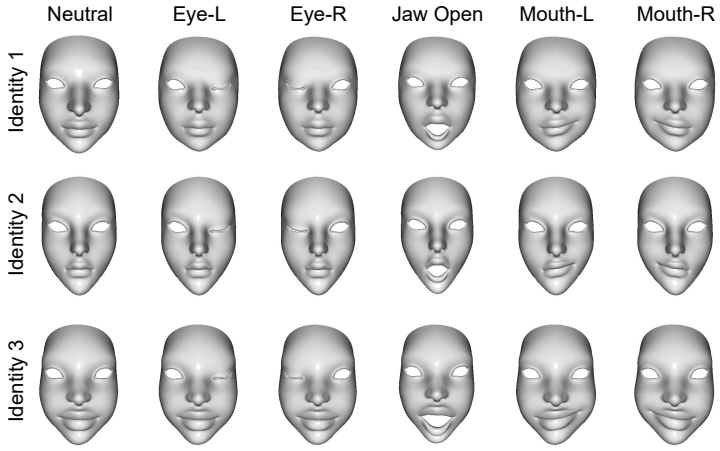


Figure 3.20: Examples of applying various expression blendshapes to stylized neutral face meshes (first column) across different facial identities.

except that we remove $\mathcal{L}_{\text{lap}}$ and $\mathcal{L}_{\text{arap}}$ since the ground-truth data consists of existing original faces. We compare reconstructions from stylized faces generated by our method and LDT. Table 3.8 reports the mean Euclidean distance between reconstructed and original meshes. The results show that our method achieves more consistent reconstruction than LDT, due to our stylization network leveraging a robust shared latent space learned on the original faces, whereas LDT processes each mesh independently. Notably, LDT achieves a lower distance on style 1, which we attribute to the style 1 template face being visually closer to the BFM face, making it easier for LDT to keep the original identity information.

Table 3.8: Reconstruction mean per-vertex Euclidean distance (measured in millimeters).

Method	**Style 1**	**Style 2**	**Style 3**	**Style 4**	**Average**
Ours	1.443	**1.441**	**1.434**	**1.425**	**1.436**
LDT [134]	**0.647**	4.544	2.986	1.823	2.5

3.4.6 Summary

We developed a neural network-based approach that is practical for 3D face shape stylization. This approach has advantages in that it does not need paired training

data of the real faces (source) and their stylized faces (target), and the topologies of the source and target meshes do not need to be identical. It is different from traditional deformation transfer methods, in that it utilizes a data-driven approach to train the neural encoder-decoder for style translation. Also, unlike existing works on learning-based face deformation transfer, our method offers the benefit of leveraging weakly supervised learning and significantly reducing the complexity and cost of preparing paired training data. An important contribution to the training process is the meticulously designed loss function that preserves the mesh structural quality and local rigidity. Our experiments demonstrate that while the results of our proposed method are comparable to those of the traditional deformation transfer in terms of quality, our method has a significant advantage in terms of speed. It is around 3,000 times faster and is thus suitable for real-time applications. In the future, we plan to explore alternative mesh regularization strategies to further enhance surface quality and enforce local geometric consistency. Automatic style template generation is also a promising future work direction.

3.5 Chapter Summary

In this chapter, we introduce two complementary approaches to advance the controllability of mesh-based 3D facial modeling. The first method, NEO-3DF, introduces an editing-oriented 3D face reconstruction framework that models the 3D face as submodels of semantically meaningful parts (such as the nose and lips). This design enables both improved 3D-to-2D alignment in the single image-based 3D face reconstruction task and intuitive face editing. The second method provides a cost-effective approach to stylize a reconstructed 3D human face to achieve entertainment effects. Traditionally, this process required 3D modeling artists to pre-design a set of templates for each style to enable the composition of a new face. This is expensive and time-consuming. Our proposed method requires only a single artist-designed template face to provide the style, and leverages the proposed weakly supervised learning algorithm to learn to translate a given 3D human face into the stylized face. The proposed methods establish a unified pipeline for reconstructing, editing, and stylizing diverse 3D faces, offering practical versatility for applications in virtual avatars and digital human creation. So far, we have

focused on the facial region of the head and relied on the traditional 3D mesh representation. In the next chapters, we will discuss how to leverage emerging 3D representations (NeRF and 3DGS) to learn 3D head creation from in-the-wild 2D face images (Chapter 4) and monocular videos (Chapter 5).

Chapter 4

3D-Aware Face Modeling

Unfortunately, no one can be told what the Matrix is. You have to see it for yourself. — Morpheus, The Matrix (1999)

In the previous chapter, we extended the mesh-based 3D Morphable Face Model (3DMM) to support intuitive editing and stylization. However, despite their explicit geometric structure and strong controllability, mesh-based models remain limited in two important ways. First, achieving a photorealistic appearance is challenging, especially for complex and flexible regions like hair and fine-scale textures. Second, the need for 3D supervision and registered mesh topology restricts their scalability to large, in-the-wild datasets, ultimately hindering generalization. To overcome these challenges, this chapter shifts to 3D-aware neural rendering, an implicit 3D representation that can infer geometry and appearance from 2D supervision alone. We also aim to retain the parametric controllability of 3DMMs, so that reconstructed heads remain editable and semantically meaningful.

4.1 Introduction

The field of 3D-aware face modeling has witnessed rapid advancements in recent years, due to its promise in many applications such as virtual reality avatars [20, 146], face editing [80, 227], and facial animation [13, 105, 116]. Unlike traditional 3D face models, which are based on 3D meshes, 3D-aware methods enable the rendering of images from controllable viewing directions without requiring ex-

plicit 3D geometry. This ability is enabled by using neural radiance fields (NeRF) models [54, 128, 184] for learning an implicit 3D face representation. The key benefits of most 3D-aware face models over traditional 3D face models are (1) they are differentiable in nature and can learn from 2D images, (2) they can represent complex structures such as hair and eyeglasses, and (3) they can produce photorealistic appearance without the need for costly 3D modeling. In recent years, 3D-aware neural rendering methods have shown promise in rendering 2D photo-realistic face images and are considered a substitute for traditional 3D textured mesh rendering [65, 71, 172]. Among traditional 3D face models, the 3D Morphable Face Model (3DMM) [14] is particularly notable for its ability to independently control identity, expression, and appearance through a parametric representation. HeadNeRF is a 3D-aware parametric face model that has been partially supervised using fitted 3DMM coefficients (codes) to achieve a disentangled control space [71]. The advantages of HeadNeRF over traditional 3DMMs are: (1) it can represent hair and teeth; (2) its rendering process is inherently differentiable; and (3) the rendered texture is more realistic than vertex color-based 3DMM.

To improve the reconstruction capability of an existing 3D-aware parametric face model, we first examine HeadNeRF and introduce a training framework that enables single-image reconstruction (Section 4.3). Despite its advantages, we identified two major issues of HeadNeRF. The first issue concerns the single-image fitting used by HeadNeRF for face reconstruction. Single-image fitting is a slow, iterative optimization process. HeadNeRF requires two stages of fitting. In the first stage, an NL3DMM [186] is fitted to derive initial codes. Since NL3DMM is a variation of 3DMM, its codes do not include non-facial parts information and cannot represent more complex textures. Thus, the initial codes need to be fine-tuned in the second stage. Both stages are time-consuming. Importantly, single-image fitting is prone to overfit the codes. For example, HeadNeRF does not represent accessories such as sunglasses. When fitting on a face with sunglasses, the optimization tends to make the area around the eyes dark and blurry to achieve a lower fitting loss. The second issue is that HeadNeRF does not have an explicit 3D shape and a fixed mesh topology. In mesh-based 3D face reconstruction, including 3DMM, using semantic face segmentation as additional supervision can enhance the reconstruction accuracy of facial part shapes [95, 216, 246]. This ap-

proach renders 3D semantic segments into 2D segments and compares them with the 2D ground-truth semantic segments using image-level loss. However, applying a similar technique in HeadNeRF is challenging.

Although our training framework improves HeadNeRF's reconstruction quality, it still exhibits limited generalization due to its reliance on HeadNeRF, which depends on large-scale lab-collected data. The training of a 3D-aware face model with a disentangled parametric space using in-the-wild face images (captured in uncontrolled, everyday environments, rather than in a structured studio setting) remains a significant challenge. To address this, we develop **StyleMorpheus** (Section 4.4), a 3D-aware face model trained solely on in-the-wild face images. Our method achieves better rendering quality and generalization ability than existing 3D-aware morphable face models [52, 71, 247]. StyleMorpheus has four main advantages. First, it can control facial identity, expression, and appearance, similar to 3DMM. Second, it only uses in-the-wild 2D images for training. Third, it does not need an explicit 3D face shape as prior. Fourth, it is lightweight and supports real-time 3D face rendering. StyleMorpheus also supports face reconstruction from a single-image input. A principal thrust of our effort is to overcome the reliance on large-scale synthetic or lab-derived data, a prevailing concern in existing 3D-aware morphable face models [52, 71, 247]. To this end, we design a novel StyleGAN-based auto-encoder, drawing inspiration from advances in StyleGAN-based face image synthesis [88], which has achieved photorealistic quality using only in-the-wild face images as training data. Our auto-encoder extracts the face-related style codes from the given face image. Following 3DMM, the style codes are separated into four semantic groups (we refer to this as the disentangled style code space) that can be controlled independently, namely identity, expression, texture, and lighting.

Together, our two methods advance 3D-aware parametric face modeling from improving an existing lab-data-trained model toward a fully in-the-wild-trained alternative with significantly improved generalization while preserving the parametric control characteristic of 3DMMs.

4.2 Related Works

4.2.1 Parametric 3D Face Modeling

Parametric 3D face modeling revolutionized the process of modeling a 3D face by using low-dimensional parameters to control the variations in the shape and appearance of the face [45, 104, 191]. The 3D Morphable Face Model (3DMM) is a type of parametric 3D face model that captures facial shape and texture variations across a population [14, 109, 139]. 3DMMs are based on principal components learned from a large dataset of 3D facial scans. This allows the model to generate a wide range of facial expressions, appearances, and identities. 3DMM usually contains four groups of parameters: identity, expression, albedo, and lighting. The identity, expression, and albedo parameters are the blending weights of their corresponding principal components, while the lighting parameters are the coefficients of the spherical harmonics (SH) lighting, used to efficiently represent varying lighting effects [167]. A popular downstream application of the 3DMM is single-image 3D face reconstruction. Its goal is to figure out the parameters that can faithfully recover the face's 3D shape and texture from the input image [38, 48, 130, 246]. Reconstruction is also the prerequisite of other applications such as user-personalized face avatars [6, 236] and 3D face editing [216].

4.2.2 NeRF-Based 3D-Aware Morphable Face Models

While 3DMM simplifies 3D face generation and enables easier control, it falls short of achieving photorealistic rendering capabilities. Also, 3DMM does not cover flexible and complex shapes such as hair and beard. On the contrary, NeRF does not need rigid 3D geometries but can synthesize 3D-consistent (i.e. 3D-aware) and photorealistic images. To achieve both controllability and realistic rendering, recent works propose to learn a 3DMM using NeRF [52, 71, 247]. MoFaNeRF is the first NeRF-based 3DMM [247]. It relies on large-scale, lab-collected datasets. These datasets are captured in controlled environments where actors perform a range of facial expressions while multiple cameras simultaneously record them from different angles. The NeRF used in MoFaNeRF is conditioned on shape, expression, and texture parameters derived from the dataset. Similar to MoFaNeRF,

HeadNeRF utilizes the large-scale lab-collected facial images as training data and uses the pre-fitted 3DMM coefficients as the initial face code to train a NeRF-based model. It further enhances the representation of the face code by fine-tuning both the face code and the model using in-the-wild face images [71]. 3DMM-RF learns on the synthetic images rendered from synthesized 3DMM faces and achieves superior generalization ability in fitting on in-the-wild face images [52]. Whereas, 3DMM-RF inherited the problem of not modeling hair from a traditional 3DMM. In contrast to the above-mentioned methods, our approach exclusively uses in-the-wild images for training, enabling greater diversity and reduced cost. This approach also allows us to effectively capture diverse hairstyles, setting our method apart from MoFaNeRF and 3DMM-RF.

4.2.3 StyleGAN-Based 3D-Aware Face Models

Generative adversarial networks (GANs) [63] have played an important role in face image generation [87]. Based on GANs, StyleGAN introduces several innovative techniques to stabilize the training process and improve the controllability of face image generation [87–89]. These include progressive training that allows photo-realism in high-resolution [86], the mapping network that enables a less entangled intermediate latent space [87], and style-based modulated convolution [88] that adjusts the statistics of intermediate feature maps to control the "style" [76] of generated image. Building upon these advancements, recent works have extended the frontier of StyleGAN-based 2D generative face models into the realm of 3D-aware synthesis. These works can be broadly grouped into two categories: methods that focus on random face image generation following the original StyleGAN paradigm [25, 106, 242, 243], and methods that aim to improve controllability and view consistency in face image generation [40, 172–174, 214, 227].

Pi-GAN [24] proposes a modulated version of the SIREN layer [166] to replace the MLP layers used in the original NeRF, resulting in a StyleGAN-based NeRF network. This allows the model to generate view-consistent face images conditioned on both the style code and camera pose. Methods such as StyleNeRF [64] and CIPS-3D [242, 243] only use the StyleGAN-based NeRF to render a low-resolution feature map, then utilize upsampling and 2D StyleGAN layers

to gradually increase the resolution. This coarse-to-fine strategy significantly reduces the computational cost. EG3D [25] presents tri-plane NeRF, which forms a tri-plane radiance field representation by reshaping the 2D feature maps generated by StyleGAN. Like the original StyleGAN, these methods randomly sample style codes to synthesize the face images, making it hard to control the generation. In our work, the style codes are disentangled through learning, allowing our generator to gain more controllability.

In recent studies, researchers have introduced 3D prior conditioned face models that combine both 3D-aware and traditional 3D representations [40, 173, 174, 214]. Among these, some methods use semantic face segmentation masks to guide face editing [172, 227], while others employ explicit 3D face geometry as a prior. For example, CGoF proposes a mesh-guided volume sampler to incorporate shape information into volume rendering [174]. OmniAvatar trains a Signed Distance Function (SDF) network to generate the head shape [214], which is then used to construct a volumetric correspondence map that guides volume rendering [84, 124]. Next3D builds on the concept of neural textures [185], synthesizing a neural texture via a StyleGAN generator [173]. This neural texture, together with a reconstructed head mesh, is used in the rasterization process to produce a texture-rasterized field for 3D-aware rendering. Utilizing a 3D prior offers the advantage of capturing shape-related deformations like facial expressions through explicit 3D geometries, which simplifies expression control. However, these approaches aren't exclusively implicit 3D rendering techniques since they still rely on explicit 3D face/head geometries during rendering, implying a reliance on accurate 3D face reconstructions. Furthermore, combining 3D geometry with 3D-aware rendering increases model complexity and impedes real-time efficiency. In contrast, our method does not use explicit 3D shapes, allowing a more efficient face rendering and simpler face reconstruction process.

4.3 Method 1: Learning Disentangled Features for NeRF-based Face Reconstruction

4.3.1 Overview

This section presents a 3D-aware face reconstruction learning framework tailored for HeadNeRF to address the two limitations identified in Section 4.1. To address the first limitation (slow, overfitting-prone single-image fitting), we propose a learning-based approach using a face encoder network. The face encoder is trained on a large face image dataset and, at inference time, directly estimates the HeadNeRF codes without the need for an iterative optimization process. To address the second limitation (the lack of explicit 3D geometry for semantic facial-parts supervision), we introduce a lightweight semantic face segmentation network, named SegLite, that renders face-parsing masks for HeadNeRF. This allows us to use facial parts-based loss functions to improve reconstruction accuracy. Our experiments show that the proposed method achieves both low reconstruction time and enhanced reconstruction accuracy. In summary, our work makes the following main contributions:

1. We propose a learning framework that trains a face encoder and uses HeadNeRF as the decoder to reconstruct face images;
2. We design a lightweight semantic face segmentation network, named SegLite, to enhance reconstruction quality without retraining HeadNeRF;
3. We introduce a facial-parts color loss function to guide the learning of facial part colors (e.g., lip color).

4.3.2 The Proposed Method

Background: HeadNeRF

The HeadNeRF model contains two sub-networks, a NeRF-based 3D-aware low-resolution *feature renderer*, and a 2D convolution-based *image renderer*. The feature renderer takes four input codes, identity code $w_{id} \in \mathbb{R}^{100}$, expression code

$w_{exp} \in \mathbb{R}^{79}$, albedo code $w_{alb} \in \mathbb{R}^{100}$, and illumination code $w_{ill} \in \mathbb{R}^{27}$, in terms of the nonlinear 3DMM (NL3DMM) [186]. A given camera pose P is used to simulate rays used in volume rendering [128]. Different from the original NeRF that renders an RGB image (usually in high-resolution), the output of HeadNeRF's feature renderer is a 256-channel 32×32 low-resolution feature map $h_{32} \in \mathbb{R}^{32\times32\times256}$. The image renderer takes h_{32} as input and gradually increases the feature maps' size to 512×512 (h_{512}). The last layer generates a rendered 512×512 RGB image $\hat{x}$ using h_{512}.

HeadNeRF uses two stages in single-image fitting-based face reconstruction. The first stage fits an NL3DMM to get the initial codes (w's) and the initial camera pose (P'). The second stage fits the HeadNeRF by finding optimal offsets Δws and ΔP to w's and P', respectively. The final codes and camera pose are computed through $w = w' + \Delta w$ and $P = P' + \Delta P$. The loss functions used in its single-image fitting task is

$$\mathcal{L}_{recon} = \mathcal{L}_{photo} + \mathcal{L}_{perc} + \mathcal{L}_{reg}, \tag{4.1}$$

where $\mathcal{L}_{photo}$ is photometric loss [38], $\mathcal{L}_{perc}$ is VGG-16 perceptual loss [81], and $\mathcal{L}_{reg} = \sum_{m\in A} ||m||^2$ (A is the set of Δws and ΔP) is a regularization loss.

The Proposed Framework

There are two main problems with single-image fitting-based face reconstruction. The first is that the fitting is an iterative, slow optimization process. The second is it's prone to overfitting. To address these problems, inspired by [38], we design a *face encoder* network that inputs a face image x and directly outputs the estimated codes $\hat{w}$s. The face encoder network uses ArcFace [37] as the backbone, which gives a 512-dimension vector. We add three fully-connected layers after ArcFace, to generate the codes. Figure 4.1 shows the proposed framework to train the face encoder.

A challenge in training the face encoder is that we do not have explicit 3D geometry information that can be used to improve the reconstruction accuracy. This differs from [38], which uses 3D mesh-based 3DMM. Therefore, we propose a segmentation network, SegLite, to render 2D segmentation masks for HeadNeRF.

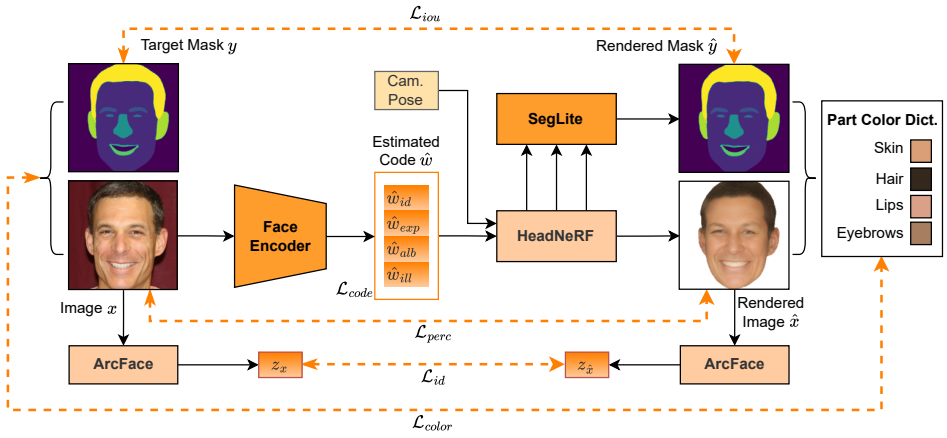


Figure 4.1: The proposed reconstruction learning framework.

Based on the rendered 2D segmentation, we can use 2D semantic-level loss functions in training. The design of SegLite reuses HeadNeRF's feature maps and is thus highly lightweight in memory usage and computation.

SegLite: Lightweight Segmentation Network

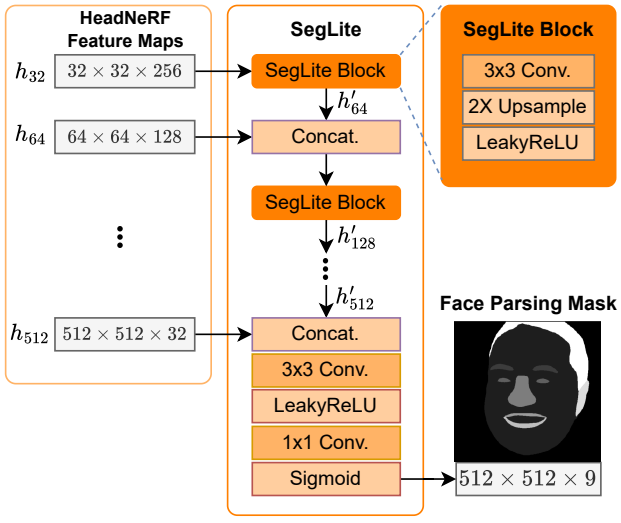


Figure 4.2: The proposed SegLite network.

We name the main building block of the proposed SegLite, the SegLite block. Each SegLite block contains a 2D convolution layer with 32 3×3 filters and a bi-linear up-sampling (2×) layer. Figure 4.2 shows the proposed SegLite. The first SegLite block takes the feature map h_{32} from HeadNeRF as input, and outputs feature map h'_{64}. Then, we concatenate (stack) h_{64} and h'_{64} and use it as the input to the second SegLite block. We repeat this process until we get the feature map h'_{512} with a resolution of 512×512. The final output of SegLite is the rendered face parsing mask $\hat{y} \in \mathbb{R}^{512\times512\times9}$ that covers nine classes: *background*, *face skin*, *eyebrows*, *eyes*, *nose*, *upper lip*, *lower lip*, *ears*, and *hair*. We use pixel-level cross-entropy loss to train SegLite.

Loss Function

This section presents our loss function to train the face encoder using our proposed framework. Denote the target parsing mask as y. We define the parts alignment loss as $\mathcal{L}_{iou} = mIoU(y, \hat{y})$, which computes the mean Intersection-over-Union (mIoU) between y and $\hat{y}$. Based on the image and mask pair, we can compute the average color of each part, which we store in a part color dictionary (see Figure 4.1). We propose the part-color loss between two part color dictionaries derived from x and $\hat{x}$ as $\mathcal{L}_{color}$, which is the sum of the L2 error between each pair of part colors. The part-based loss is defined as

$$\mathcal{L}_{part} = \mathcal{L}_{iou} + \mathcal{L}_{color}. \tag{4.2}$$

Inspired by [34], to ensure the reconstructed face and the face on the target image look like the same person, we define identity loss as

$$\mathcal{L}_{id} = ||z_x - z_{\hat{x}}||^2, \tag{4.3}$$

where z_x and $z_{\hat{x}}$ are extracted deep facial identity feature vectors using a pre-trained ArcFace network [37]. In addition, we define the face code loss based on

[71] as

$$\begin{aligned}\mathcal{L}_{code} = ||\hat{w}_{id} - w_{id}||^2 + 10 \times ||\hat{w}_{exp} - w_{exp}||^2 \\ + ||\hat{w}_{alb} - w_{alb}||^2 + ||\hat{w}_{ill} - w_{ill}||^2,\end{aligned} \tag{4.4}$$

where ws are pre-fitted NL3DMM codes. The final loss function used in training the face encoder is defined as

$$\mathcal{L} = \lambda_{iou}\mathcal{L}_{iou} + \lambda_{code}\mathcal{L}_{code} + \lambda_{perc}\mathcal{L}_{perc} + \lambda_{id}\mathcal{L}_{id} + \lambda_{color}\mathcal{L}_{color}, \tag{4.5}$$

where λs are loss term weights.

4.3.3 Experiments

Dataset

In this work, we utilize the Flickr-Faces-HQ (FFHQ) dataset [87], which contains 70,000 face images with a resolution of 1024×1024. To match the size of the rendered images produced by HeadNeRF, we resize all images to 512×512 using bi-linear interpolation for both training and evaluation purposes. We generate target face-parsing masks for the dataset using an off-the-shelf face-parsing network [1]. We randomly select 1,500 images for testing and use the remaining images for training.

Implementation Details

To train SegLite, we use synthesized training data. Here we use the fitted HeadNeRF codes to render the RGB face images. Then we run the face-parsing network to generate target parsing masks. We use an initial learning rate of 10^{-3} that linearly decreases to 0.5×10^{-5} over the training duration. We train SegLite for 10 epochs.

We train the face encoder in two stages. In the first stage, we only use $\mathcal{L}_{code}$ in our loss function. We first load the pre-trained ArcFace weights and train the

[1] github.com/zllrunning/face-parsing.PyTorch

newly-added layers for 5 epochs. Then, we train the entire face encoder for 15 epochs. The initial learning rate is 10^{-3}. In the second stage, we use $\mathcal{L}$ as the loss function to fine-tune the face encoder for 50 epochs. The loss term weights are set to: $\lambda_{iou} = 0.1$, $\lambda_{code} = 1$, $\lambda_{perc} = 0.001$, $\lambda_{id} = 0.01$, and $\lambda_{color} = 20$. The initial learning rate is 10^{-5}, and the final learning rate decreases to 10^{-6}. We run single-image fitting on our dataset to generate the target codes ws used in computing $\mathcal{L}_{code}$. We perform 100 optimization iterations in each fitting. We use $\mathcal{L}_{recon} + \mathcal{L}_{part}$ as the improved fitting loss function.

All experiments were conducted on a cloud computing instance with one NVIDIA A100 (SXM4 40GB) GPU. We use Adam optimizer for optimization. The batch size is set to 32 in both SegLite and face encoder training. For HeadNeRF, we set the number of sampled points along each ray to 32.

Comparison

We compare our face encoder-based reconstruction with single-image fitting-based reconstruction. In single-image fitting, the baseline uses the original HeadNeRF fitting loss $\mathcal{L}_{recon}$. To demonstrate the importance of the proposed SegLite-enabled loss $\mathcal{L}_{part}$, we also experiment with using $\mathcal{L}_{recon} + \mathcal{L}_{part}$ as the improved fitting loss (namely, HeadNeRF + SegLite).

We follow [34] to use deep facial identity features in evaluating the reconstruction accuracy in terms of the similarity of facial identities. The approach uses a deep face image encoder network, pre-trained on face image datasets such as VGG Face [21], to separately encode the target face image and the reconstructed face image and compute the L2 error. In our experiment, we use pre-trained FaceNet [155], and ArcFace [37]. The results are shown in Table 4.1. Our method achieves a lower error and a lower standard deviation (more stable) compared with HeadNeRF's fitted results.

Table 4.1: Quantitative comparison of reconstruction.

Method	**FaceNet L2 Error↓**	**ArcFace L2 Error↓**
HeadNeRF	1.310 ($\pm$ 0.338)	713.9 ($\pm$ 143.1)
HeadNeRF + SegLite	1.193 ($\pm$ 0.342)	696.8 ($\pm$ 143.8)
Ours	**1.085 ($\pm$ 0.319)**	**635.4 ($\pm$ 133.2)**

In Figure 4.3, we visually compare the reconstruction results of HeadNeRF's original fitting method (second row), fitting with our SegLite-enabled loss $\mathcal{L}_{\text{part}}$ (third row), and reconstruction using our proposed face encoder (last row). Column 6 illustrates that using the proposed part-based loss leads to more accurate color reconstruction for facial regions. The single-image fitting approach often produces reconstruction artifacts (e.g., blurry faces) when input images contain occlusions, since the training data used by HeadNeRF does not include such cases. For example, artifacts occur when the person wears eyeglasses (columns 1 and 3) or other accessories such as a hat (columns 7 and 8). Despite these limitations, our trained encoder demonstrates greater robustness under these challenging conditions.



Figure 4.3: Example qualitative face reconstruction results. The first row shows the ground-truth (target) images. The second row presents reconstructed face images using HeadNeRF's original fitting method. In the third row, the proposed $\mathcal{L}_{\text{part}}$ is included in HeadNeRF's fitting process as an additional regularization loss. The last row shows reconstructed face images via our proposed framework.

Table 4.2 compares the time used in face reconstruction process. The second row shows that using SegLite enabled $\mathcal{L}_{part}$ as additional regularization loss in single-image fitting only slightly increased the running time. The last row shows that our face encoder can significantly reduce the reconstruction time compared with the single-image fitting.

We use an ablation study to quantitatively justify that the proposed part-based loss term improves single-image fitting accuracy by using traditional image quality

Table 4.2: Per image fitting time comparison. Results are shown in seconds. **N**: NL3DMM fitting; **H**: HeadNeRF fitting; **C**: camera pose estimation; **E**: encoder inference.

Method	**N**	**H**	**C**	**E**	**Total↓**
HeadNeRF	3.811	5.612	N/A	N/A	9.423
HeadNeRF + SegLite	3.811	5.734	N/A	N/A	9.545
Ours	N/A	N/A	1.325	0.017	**1.342**

evaluation metrics. Results are shown in Table 4.3.

Table 4.3: Ablation study results on proposed part-based loss term.

Fitting Loss	**L1 ↓**	**PSNR ↑**	**SSIM ↑**
$\mathcal{L}_{recon}$	0.229 (±0.078)	17.75 (±2.669)	0.803 (±0.047)
$\mathcal{L}_{recon} + \mathcal{L}_{part}$	**0.186 (±0.061)**	**19.17 (±2.519)**	**0.814 (±0.045)**

4.3.4 Summary

This section presents a novel approach for reconstructing 3D-aware faces that is computationally efficient and generates high-quality and accurate reconstructed faces. Our approach uses a face encoder network to directly estimate the codes used by HeadNeRF to render a face image. To better train the face encoder, we proposed the SegLite network that allows us to use the semantic facial parts-based loss as additional regularization. Our experimental results show that the proposed face encoder achieves superior reconstruction accuracy while significantly reducing the running time. However, our method also inherits certain limitations from the underlying HeadNeRF model, such as limited ability to handle occlusions, restricted generalization to unseen identities, and the absence of fine-grained appearance details. To overcome these limitations, in the next section (Section 4.4), we present our 3D-aware morphable face model, StyleMorpheus.

4.4 Method 2: A StyleGAN-Based 3D-Aware Morphable Face Model with a Disentangled Style Space (StyleMorpheus)

4.4.1 Overview

Recent advances in 3D-aware neural rendering have enabled photorealistic face image synthesis from arbitrary viewpoints. However, achieving disentangled control over facial attributes typically depends on large, curated datasets collected in controlled environments. To overcome this limitation, we introduce ***StyleMorpheus***, a 3D-aware, StyleGAN-based morphable face model that can be trained entirely on in-the-wild 2D face images. Our method achieves better rendering quality and generalization ability than existing 3D-aware morphable face models [52, 71, 247]. Unlike conventional StyleGAN-based methods, StyleMorpheus also provides disentangled control over facial identity, expression, and appearance, allowing each attribute to be adjusted independently without unintended changes to the others.

StyleMorpheus has four main advantages. First, it can control facial identity, expression, and appearance, similar to 3DMM. Second, it only uses in-the-wild 2D images for training. Third, it does not need an explicit 3D face shape as prior. Fourth, it is lightweight and supports real-time 3D face rendering. StyleMorpheus also supports face reconstruction from a single-image input. To this end, we design a novel StyleGAN-based auto-encoder. Our encoder extracts the face-related style codes from the given face image and learns a representative, disentangled style code space. Following 3DMM, the style codes are separated into four semantic groups (we refer to this as the disentangled style code space) that can be controlled independently, namely identity (id.), expression (expr.), texture (tex.), and lighting (light). Our decoder is designed for real-time efficiency, anchored upon a low-resolution NeRF and 2D neural rendering, and enforces disentanglement by using shape- and appearance-related codes at different levels of the network. We utilize modulated layers, similar to those in StyleGAN, to enable the decoder to apply the desired styles encoded in the style code to the feature maps, thereby controlling the generated face. Our training strategy has two stages: in the first stage, we train the auto-encoder to learn a representative latent style code space; in the second stage,

we fine-tune the decoder with additional adversarial loss to improve the image quality. We further demonstrate the disentanglement capabilities of StyleMorpheus through face editing tasks such as style mixing, face morphing, and color editing.

The main contributions of this work are:

1. We propose StyleMorpheus, a novel 3D-aware morphable face model together with an effective training strategy. StyleMorpheus learns solely from in-the-wild 2D face images, eliminating the need for controlled, lab-collected datasets. It also achieves superior reconstruction and rendering quality compared to existing 3D-aware morphable face models;

2. While inspired by StyleGAN, StyleMorpheus introduces a disentangled style code space, unlike existing StyleGAN-based methods. This enables independent control over facial attributes such as identity, expression, and appearance;

3. We demonstrate the disentangled control ability of StyleMorpheus through various applications, including style mixing, face morphing, and color editing of facial parts.

4.4.2 The Proposed Method

Model Architecture

The representation ability of the original 3DMM codes is limited mainly by two reasons. First, the 3DMM does not model the hair and teeth due to the complexity of modeling them in 3D. Second, the vertex color-based texture model is low-resolution, which cannot accurately represent complex textures such as facial hair. Therefore, we design an auto-encoder architecture (see Figure 4.4) to enable learning a more representative code space. We use the pre-trained ArcFace [37] as the backbone of our face encoder. We connect four MLP-based networks to the face encoder, as semantic heads to derive the disentangled style space $\mathcal{Z} = \{z_{id} \in \mathbb{R}^{100}, z_{expr} \in \mathbb{R}^{79}, z_{tex} \in \mathbb{R}^{100}, z_{light} \in \mathbb{R}^{27}\}$. The dimensions of our style codes $z \in \mathcal{Z}$ align with nonlinear 3DMM [186]. Different from the StyleGAN's $\mathcal{Z}$ space which is a unit normal distribution $\mathcal{N}(0, 1)$, our $\mathcal{Z}$ space is learned to be

disentangled. Following StyleGAN, we use mapping networks to map the style code from $\mathcal{Z}$ space to $\mathcal{W}$ space. The dimensions of $w \in \mathcal{W}$ are the same as $z \in \mathcal{Z}$.

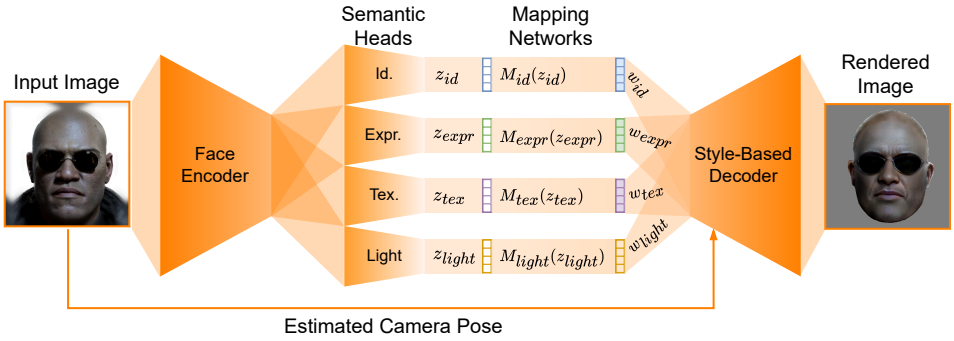


Figure 4.4: The proposed model architecture follows an auto-encoder structure. The encoder consists of an ArcFace-based face image encoder and four semantic heads to produce identity, expression, texture, and lighting codes in $\mathcal{Z}$ space. Following the semantic heads are mapping networks Ms that map the corresponding style code to ws in $\mathcal{W}$ space. The style codes in $\mathcal{W}$ space are then used in the proposed StyleGAN-based decoder to reconstruct the input image.

Our StyleGAN-based decoder follows a coarse-to-fine strategy to achieve real-time rendering [64, 71]: as NeRF suffers from high computational cost in rendering high-resolution images, we only use a NeRF-based renderer to render the low-resolution feature map $I_f \in \mathbb{R}^{32\times32\times d_f}$ at 32×32 resolution, where d_f is the number of feature channels. We use $d_f = 256$ in our experiment. Then we apply four render blocks to generate the RGB image $\hat{I}_{512} \in \mathbb{R}^{512\times512\times3}$ at 512×512 resolution. We implement the modulated layer by using the weight-demodulated 2D convolution layer [88] with 1×1 convolution filters. It is demonstrated that 1×1 convolution layer is rotation-and-shift equivariant which is important in the 3D-aware setting [64, 89]. We use the modulated layer in both NeRF and render blocks to allow them to accept style codes.

We formulate the camera pose as $p \in \mathbb{R}^6$, where the first three values are yaw, pitch, and roll angles for the 3D rotation $\mathbf{R} \in SO(3)$ and the last three values represent the 3D translation $\mathbf{t} \in \mathbb{R}^3$. Our camera model uses perspective projection.

Given a camera pose p we march 32^2 rays through a 32×32 pixel grids and sample N points along each ray. Positional encoding is used to map the 3-dimensional point locations to $d_p > 3$ dimensions to capture high-frequency content [128]. When applying our modulated layer on the point features, we reshape the point features to $32^2 \times N \times d_p$ so that the 1×1 2D convolution acts like a fully connected operation "channel-wise". The modulated layers in our NeRF use the sine activation function following SIREN [166], to improve the ability to model high-frequency information. Since the NeRF is low-resolution, we only expect it to learn shape-related representation. For this reason, we only apply w_{id} and w_{expr} to our NeRF's modulated layers. The outputs of our NeRF are the estimated densities $\sigma \in \mathbb{R}^{32\times32\times N}$ and features $r \in \mathbb{R}^{32\times32\times N\times d_f}$ of the sampled points. The feature map I_f is rendered through volume rendering [84, 124]. For clarity, we reformulate the volume rendering equation in the following discrete format:

$$I_f(x, y, k) = \sum_{i=1}^{N-1} t(x, y, i)\alpha(x, y, i)r(x, y, i, k), \tag{4.6}$$

where $t(x, y, i)$ is the accumulated transmittance from the first point to the i^{th} point along the ray originating from the camera and passes through the pixel location at (x, y), $\alpha(x, y, i)$ is the corresponding alpha compositing value, and k indicates the k^{th} feature channel ($1 \leq k \leq d_f$). Theoretically, one needs to ensure $N >> 1$ when sampling the points. Since Equation 4.6 is an approximation of the continuous volume rendering function, the larger sampling size N leads to a more accurate approximation. Equations 4.7 and 4.8 are the computations of $t(x, y, i)$ and $\alpha(x, y, i)$, where $\delta(x, y, i)$ represents the distance between the i^{th} and $(i+1)^{th}$ adjacent points on the ray passes through (x, y).

$$t(x, y, i) = \begin{cases} exp[-\sum_{j=1}^{i-1} \sigma(x, y, j)\delta(x, y, j)] & \text{if } i > 1, \\ 1.0 & \text{if } i = 1. \end{cases} \tag{4.7}$$

$$\alpha(x, y, i) = 1 - exp[-\sigma(x, y, i)\delta(x, y, i)]. \tag{4.8}$$

We use four render blocks to increase the resolution and control the texture and lighting effects of the final rendered image. In each render block, we start

with a learnable $2\times$ upsampling layer based on PixelShuffle [162] and a fixed blur kernel [233]. Followed by the upsampled feature map are two modulated layers, and we use LeakyReLU as the activation function. Each render block has an attached 1×1 2D convolution layer to convert the layer's output to an RGB image $\hat{I}_{res}$ (res indicates the corresponding render block's resolution). We apply light-weighted SegLite blocks [217] that utilize the feature maps from different rendered resolutions, to generate the binary head mask $\hat{\mathbf{M}}_{512} \in \mathbb{R}^{512\times512}$. Please refer to Figure 4.5 for our detailed decoder architecture.

Loss Function

The proposed framework mainly learns to reconstruct the input face image. Following deep learning single-image face reconstruction methods [38, 48, 112], we use photometric loss and perceptual loss as our reconstruction loss. The photometric loss is also known as pixel loss, which is the averaged pixel-level distances between the reconstructed and target images. Inspired by progressive training [86], we compute L1 distance at different resolution scales from 64×64 up to 512×512 to improve the convergence speed and encourage each render block to learn sufficient information. Denote I as the ground-truth image with resolution greater than or equal to 512×512, and its corresponding head mask as $\mathbf{M}$ (including face, hair, and ears), our multi-resolution photometric loss is defined as:

$$\mathcal{L}_{photo} = \sum_{i=log_2 64}^{log_2 512} \gamma_{2^i} \| \hat{I}_{2^i} - \mathbf{M}_{2^i} \odot I_{2^i} \|_1, \tag{4.9}$$

where γs are the weighting factors, $\odot$ is the Hadamard product. We use bi-cubic interpolation to down-scale I and $\mathbf{M}$ to match the resolution of each render block.

The perceptual loss captures the semantic content of images, encouraging generated images to exhibit similar high-level features to those of the target images. We use LPIPS loss [234] as our perceptual loss $\mathcal{L}_{perc}$ between $\hat{I}_{512}$ and $\mathbf{M}_{512} \odot I_{512}$. The LPIPS loss implementation we use is from PerceptualSimilarity[2], and we use its pre-trained VGG-16 network [165] as the feature extractor. We also tested the identity loss [38, 52] in our experiment. Its effect is limited in our setting; thus,

[2] https://github.com/richzhang/PerceptualSimilarity

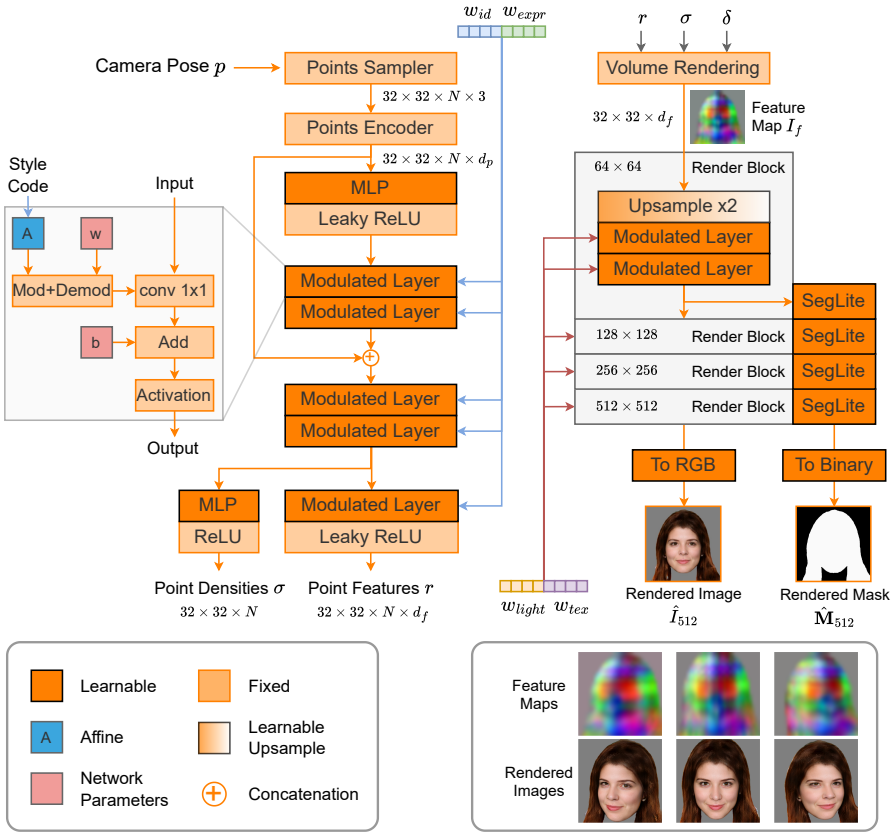


Figure 4.5: The proposed StyleGAN-based decoder architecture comprises a low-resolution NeRF that produces the 3D-aware feature map (left) and a set of render blocks (right) that generate the high-resolution RGB face image. For convenience, we only show the last "To RGB" layer in this figure. We use modulated layers in both NeRF and render blocks to accept style codes. The identity and expression codes are concatenated to input to the NeRF's modulated layers. The texture and lighting codes are concatenated and used in the render blocks to control the appearance of the generated face.

we do not use identity loss in our reconstruction loss. Our reconstruction loss is defined as:

$$\mathcal{L}_{recon} = \lambda_{photo}\mathcal{L}_{photo} + \lambda_{perc}\mathcal{L}_{perc}, \tag{4.10}$$

where λs are loss term weights.

The reconstruction loss only encourages the decoder to generate the face image with a black background. We use L_{seg} which is the averaged pixel-level cross-entropy loss between $\hat{\mathbf{M}}_{512}$ and $\mathbf{M}_{512}$ to train the light-weighted SegLite-based segmentation branch. The SegLite-based segmentation branch is useful in removing the rendered background without affecting real-time efficiency.

Following [71], we compute the code distance between zs and the estimated non-linear 3DMM [186] coefficients cs, to ensure the code space $\mathcal{Z}$ is disentangled and aligned with 3DMM. This code loss term is used as a regularization:

$$\mathcal{L}_{code} = \sum_{g} \lambda_g \| z_g - c_g \|_2^2, \tag{4.11}$$

where $g \in \{id, expr, tex, light\}$, and λs are the corresponding loss term weights.

Because in-the-wild images have been used as training data, we can only estimate the camera pose for each image. The estimation is done by fitting a 3DMM face to the image and using the optimized camera pose. In this case, we assume the camera intrinsics are the same across all the images. However, in reality, in-the-wild images are typically captured by cameras with varying optical configurations. Moreover, fitting a 3DMM to a single image also encounters the unavoidable problem of inaccurate reconstruction. Therefore, the camera poses are inaccurate for in-the-wild images, which causes the learned 3D-aware model to generate blurry images. To address this problem, we utilize adversarial learning to enhance the generation quality of our StyleGAN-based decoder and to encourage the decoder to learn how to generate high-frequency details. Denote E as our face image encoder, G as the generator (including mapping networks and the StyleGAN-based decoder), and D as the discriminator network. In terms of adversarial learning [63], our objective is to solve the following minimax game:

$$\min_{G} \max_{D} \mathop{\mathbb{E}}_{(I,p)\sim \text{data}} \{ logD(I) + log[1 - D(G(E(I), p))] \}, \tag{4.12}$$

where p is the estimated camera pose of I. Based on Equation 4.12, the discrimi-

nator loss is defined as:

$$\begin{aligned}\mathcal{L}_D = &log(1 + exp(D(G(E(I), p)))) \\ &+ log(1 + exp(-D(I))) + \mathcal{L}_{R1},\end{aligned} \tag{4.13}$$

where $\mathcal{L}_{R1}$ is the R1 regularization loss [127], which penalizes the discriminator's gradient on input images to encourage smoothness in the discriminator's output, thus improving training stability. The generator loss is defined as:

$$\mathcal{L}_G = log(1 + exp(-D(G(E(I), p)))). \tag{4.14}$$

Training Strategy

Our training strategy has two stages. In the first stage, we train the auto-encoder end-to-end. Because the face encoder is a pre-trained ArcFace network, in the first training epoch, we freeze the face encoder (not including the semantic heads) and train the rest of the network. Starting from the second training epoch, we train the entire network. The objective of this stage is:

$$\underset{\theta_E, \theta_G}{argmin}\ \mathcal{L}_{recon} + \mathcal{L}_{code} + \lambda_{seg}\mathcal{L}_{seg}, \tag{4.15}$$

where θ represents the network parameters. Note that, we do not use the segmentation loss term $\mathcal{L}_{seg}$ until the reconstruction stabilizes.

In the second stage, we fine-tune the generator with adversarial loss. We train the generator and discriminator alternately in the same way as StyleGAN2 [88]. Differently, when training the generator, we use reconstruction loss, segmentation loss, and generator loss:

$$\underset{\theta_G}{argmin}\ \mathcal{L}_{recon} + \lambda_{seg}\mathcal{L}_{seg} + \lambda_G\mathcal{L}_G. \tag{4.16}$$

We progressively fine-tune the generator following [86], starting from the 64×64 render block, to improve training stability and convergence.

Fitting-based Single-Image Face Reconstruction

Since our model is naturally differentiable, it supports single-image face reconstruction by optimizing (fitting) the style codes zs. Given an image I, we first use our face encoder to estimate the style code $E(I) = \{z_{id}, z_{expr}, z_{tex}, z_{light}\}$ as the initial code. The objective is to optimize the code offsets Δz_g to allow the rendered face image to be as close to the target face image as possible:

$$\underset{\{\Delta z_g\}}{argmin}\ \mathcal{L}_{recon}(G(\{z'_g | z'_g = z_g + \Delta z_g\}, p), I), \tag{4.17}$$

where $g \in \{id, expr, tex, light\}$, and p is the estimated camera pose. Note that the camera poses used during training are estimated by fitting the 3DMM to the images. In fitting-based single-image face reconstruction, we use an off-the-shelf face pose estimation network, 6DRepNet [69], to estimate the yaw, pitch, and roll angles of the face. Because in our framework, we assume the 3D faces are in the canonical pose and the input images are aligned, we can find a transformation to convert the estimated face pose $SO(3)$ to our camera pose $p \in \mathbb{R}^6$. We use gradient descent to optimize the code offsets. The fitting loss is defined as:

$$\mathcal{L}_{fit} = \mathcal{L}_{recon} + \sum_g \|\Delta z_g\|_2^2, \tag{4.18}$$

where $\sum_g \|\Delta z_g\|_2^2$ is a regularization term to prevent overfitting on a single image. Note that the fitting only needs to be done once on a given image, and then we can render the reconstructed face with varying views in real-time.

4.4.3 Experiments

Dataset

In this work, we mainly use the Flickr-Faces-HQ (FFHQ) dataset [87]. The FFHQ dataset contains 70,000 in-the-wild face images with a resolution of 1024×1024. We use an off-the-shelf face-parsing model[3] to generate the head masks for FFHQ. We use the first 1,000 images as test data, the following 1,000 images for vali-

[3] github.com/zllrunning/face-parsing.PyTorch

dation, and the remaining images for training. The training set contains 66,422 images (rather than 68,000), as some images cannot be successfully fitted with the 3DMM due to issues such as extreme poses. We also use the CelebAMask-HQ dataset [102] to test the generalizability of our model. The CelebAMask-HQ dataset comprises 30,000 in-the-wild face images of various celebrities, each with a resolution of 512×512. It also comes with manually annotated face-parsing masks, which we use to derive the binary head masks. We use the same set of 550 randomly selected images for testing as in [52].

Implementation Details

In the multi-resolution photometric loss term, we configure the weighting factors as follows: $\lambda_{64} = 0.01$, $\lambda_{128} = 0.1$, $\lambda_{256} = 0.5$, and $\lambda_{512} = 1.0$. For the code loss, we employ weights of $\lambda_{id} = 0.001$, $\lambda_{expr} = 0.1$, $\lambda_{tex} = 0.001$, and $\lambda_{light} = 0.01$. The remaining loss term weights are assigned as: $\lambda_{photo} = 1.0$, $\lambda_{perc} = 0.1$, $\lambda_{seg} = 0.01$, and $\lambda_G = 0.1$.

The input images for our encoder adhere to FFHQ's alignment procedure[4], ensuring consistent alignment. Our discriminator network is identical to that in StyleGAN2 [88]. The images are stored and displayed in the 8-bit RGB format. Both input and target images for our network are normalized to fit the intensity range of $[-1.0, 1.0]$. During evaluation, pixel intensities are normalized to the range of $[0, 1.0]$.

In the first training stage, we use an effective batch size of 32 through the gradient accumulation trick (the actual batch size is 8). The number of sampled points along each ray is set to 32 ($N = 32$). We use Adam optimizer with an initial learning rate of 10^{-4} to optimize the network parameters. Following each training epoch, the learning rate is reduced by 2%. In the second training stage, we use $N = 64$. The batch size is set to 32 when the resolution is less than 256×256, 16 when the resolution is 256×256, and 12 when the resolution is 512×512. We follow StyleGAN2 and use a learning rate of 10^{-4} for discriminator training and 2×10^{-3} for generator training. We conduct the training on one NVIDIA A6000 GPU. The first training stage takes 5 days, and the second training stage takes 6

[4] https://github.com/NVlabs/ffhq-dataset/blob/master/download_ffhq.py

days.

In single-image fitting, we use the Adam optimizer with an initial learning rate of 0.01 and a decay rate of 0.99 per optimization step. We optimize for 200 steps on each image.

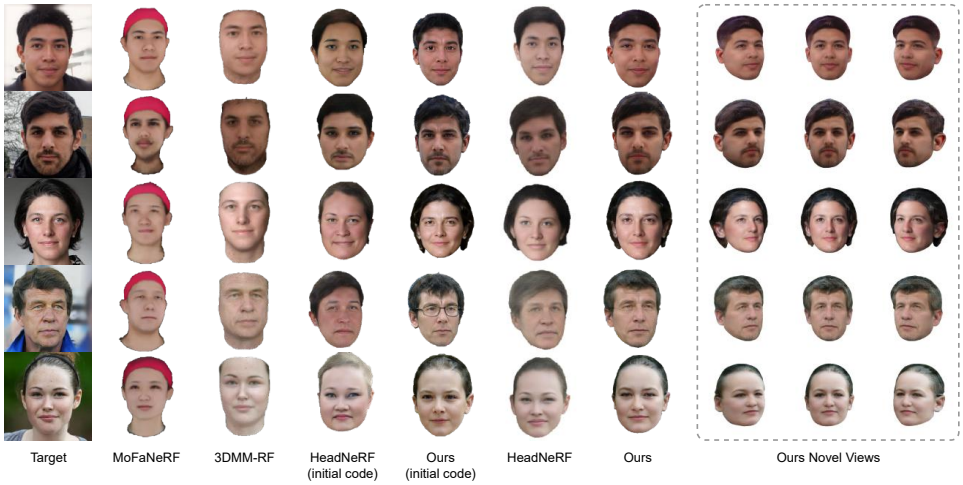


Figure 4.6: Single image face reconstruction results. The leftmost column shows the target images. We compare with MoFaNeRF [247], 3DMM-RF [52], and HeadNeRF [71]. HeadNeRF (initial code) represents the HeadNeRF's reconstructed face using fitted 3DMM code. Ours (initial code) represents our face reconstruction with our face encoder's estimated style code. The right side shows our reconstructed face rendered in different camera views.

Single-Image Face Reconstruction

We evaluate our model on the single-image face reconstruction task to assess its ability to accurately infer facial structure and texture from a single input image. For a fair comparison, we benchmark our method against existing 3D-aware morphable face models: MoFaNeRF [247], HeadNeRF [71], and 3DMM-RF [52]. Similar to HeadNeRF, which uses fitted 3DMM coefficients as the initial latent code, our approach employs the face encoder's estimated code as initialization. Accordingly, we include comparisons with both HeadNeRF's initial and optimized results.

In Figure 4.6, we demonstrate the qualitative face reconstruction results on

five images collected from the Internet. Confined by the training datasets used in MoFaNeRF and 3DMM-RF, both methods can only render parts of the face: MoFaNeRF does not render the hair, and 3DMM-RF does not render both hair and ears. Because the 3DMM only models the facial part, its coefficients cannot represent the hair. Thus, HeadNeRF cannot accurately reconstruct the hair using the 3DMM-based initial code. In contrast, our initial code allows our model to render the face close to the target, providing a good starting point for further fine-tuning the code. With the help of adversarial learning, our model can generate more photorealistic details than the other methods. We observe some side effects of adversarial learning in our case. For example, in the fourth row of Figure 4.6, our reconstructed face using the initial code has eyeglasses, different from the target image. We interpret this phenomenon as caused by the imperfect disentanglement in GANs [30]. We can solve it with fine-tuning of the code, and the eyeglasses disappear; the final result looks closer to the target image. In Figure 4.7, we use more examples to demonstrate the novel view rendering ability of our model.

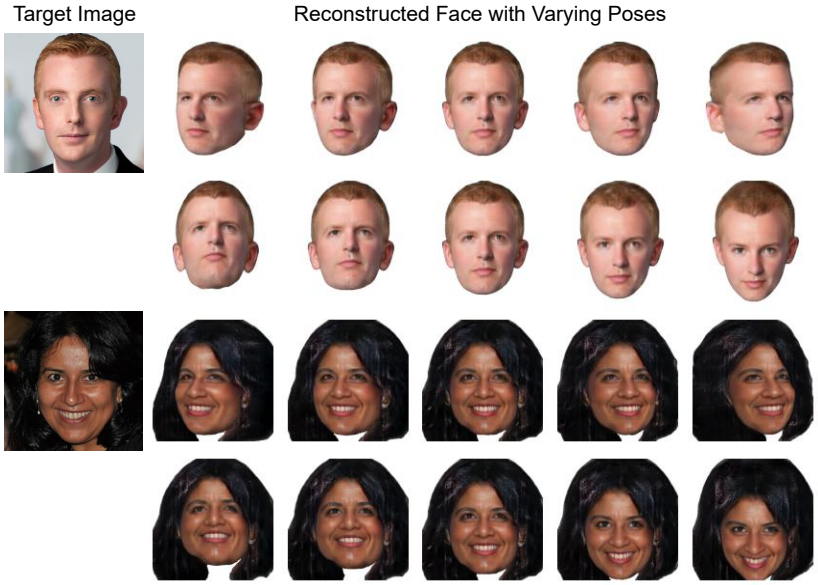


Figure 4.7: Novel view synthesis of reconstructed faces.

We also evaluate the methods quantitatively, through image-level metrics: L1

distance, LPIPS distance [234], structural similarity index (SSIM), and peak signal-to-noise ratio (PSNR). Table. 4.4 and 4.5 show the evaluation on CelebAMask-HQ dataset [102]; Table. 4.6 and 4.7 show the evaluation on FFHQ dataset [87]. Our method achieves superior reconstruction performance in terms of image-level similarity.

Table 4.4: Comparisons in reconstruction performance (Facial Region Only). Evaluated on the CelebAMask-HQ dataset.

Method	**L1** ↓	**LPIPS** ↓	**SSIM** ↑	**PSNR** ↑
MoFaNeRF	0.273	0.442	0.910	14.713
3DMM-RF*	0.216	-	**0.956**	-
HeadNeRF (initial code)	0.086	0.133	0.928	23.441
HeadNeRF	0.072	0.113	0.940	24.937
Ours (initial code)	0.068	0.112	0.927	24.619
Ours	**0.048**	**0.094**	0.951	**27.586**

* indicates results reported from the original paper (not open-source).
- indicates the value is missing or not comparable.

Table 4.5: Comparisons in reconstruction performance (Including Hair). Evaluated on the CelebAMask-HQ dataset.

Method	**L1** ↓	**LPIPS** ↓	**SSIM** ↑	**PSNR** ↑
HeadNeRF (initial code)	0.366	0.472	0.675	14.926
HeadNeRF	0.354	0.431	0.693	15.429
Ours (initial code)	0.208	0.355	0.702	18.619
Ours	**0.161**	**0.313**	**0.742**	**21.041**

Table 4.6: Comparisons in reconstruction performance (Facial Region Only). Evaluated on FFHQ dataset.

Method	**L1** ↓	**LPIPS** ↓	**SSIM** ↑	**PSNR** ↑
MoFaNeRF	0.251	0.416	0.720	15.084
HeadNeRF (initial code)	0.106	0.171	0.902	22.211
HeadNeRF	0.087	0.151	0.920	23.858
Ours (initial code)	0.085	0.152	0.900	23.533
Ours	**0.063**	**0.124**	**0.922**	**26.038**

Table 4.7: Comparisons in reconstruction performance (Including Hair). Evaluated on FFHQ dataset.

Method	**L1** ↓	**LPIPS** ↓	**SSIM** ↑	**PSNR** ↑
HeadNeRF (initial code)	0.241	0.355	0.774	17.499
HeadNeRF	0.211	0.310	0.798	18.794
Ours (initial code)	0.160	0.281	0.781	20.254
Ours	**0.126**	**0.257**	**0.814**	**22.269**

Quantitative Analysis of the Disentangled Code Space

In this experiment, we quantitatively evaluate the disentanglement capability of our latent code space in comparison to that of HeadNeRF. Disentanglement aims to enable independent control over individual features (e.g., expression) without unintentionally affecting others (e.g., texture). To assess this, we conduct a code replacement experiment: for each of the four feature sets (identity, expression, texture, and lighting), we replace one set of codes in a source image with those from a target image, while keeping the remaining three unchanged.

To evaluate the effect of this manipulation, we use the 3DMM-based face reconstruction network DECA [48] to estimate the four sets of 3DMM coefficients for (1) the source image, (2) the target image, and (3) the rendered image generated using the mixed codes. Ideally, the estimated 3DMM coefficients of the non-replaced features in the rendered image should be close to those of the source image. In contrast, the coefficients of the replaced feature should resemble those of the target image.

We conduct this experiment using 500 source-target image pairs sampled from our FFHQ test set: the first 500 images are used as sources, and the last 500 as targets. Figure 4.8 shows the mean squared error (MSE) between the estimated 3DMM coefficients of the rendered images and those of the source images (non-replaced features). Our latent code space consistently achieves lower MSE than HeadNeRF across most cases. Figure 4.9 shows the MSE between the estimated coefficients (of the replaced feature only) of the rendered images and those of the target images. Again, our model outperforms HeadNeRF, with especially notable improvements in expression and texture. It is worth noting that HeadNeRF is trained on a dataset with controlled features, while our model is trained solely on in-the-wild images without any manual labeling. Despite this, our method not only generalizes better but also demonstrates superior disentanglement in most aspects.

Quantitative Analysis of the Code Representation Ability

In this experiment, we quantitatively evaluate the quality of representation in our learned code space. In generative models, it is typically desirable for each latent code dimension to follow a standard normal distribution. This facilitates smooth

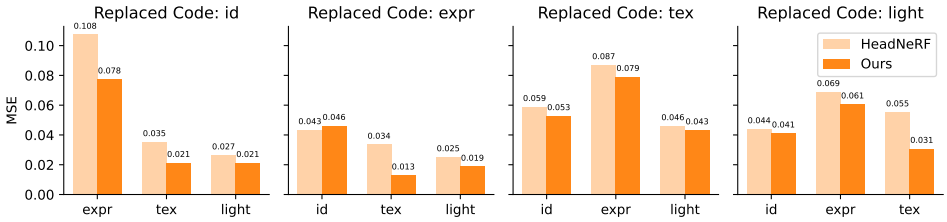


Figure 4.8: Disentanglement evaluation through code replacement. Each subplot corresponds to replacing one set of codes (identity, expression, texture, or lighting) while keeping the other three fixed. We measure the MSE between the estimated 3DMM coefficients of the rendered image and those of the original source image (i.e., for the non-replaced features). Lower MSE indicates better disentanglement.

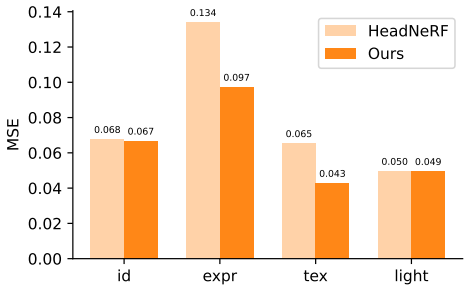


Figure 4.9: Similarity of the replaced feature to the target image. This plot shows the MSE between the estimated 3DMM coefficients of the rendered image and those of the target image (for the replaced feature). Lower MSE indicates better code controllability. Our method achieves better target similarity, especially in expression and texture codes.

interpolation and disentanglement of each dimension, which indicates a better generalization ability. We conduct this analysis by collecting the estimated 3DMM codes, the fitted HeadNeRF codes, and the codes produced by our method across 1,000 test images in our FFHQ test set. For each dimension, we apply z-score normalization, then compute the mean Kullback–Leibler (KL) divergence from the standard normal distribution.

Figure 4.10 visualizes the distributions of the first five dimensions. Notably, both 3DMM and HeadNeRF often exhibit multi-modal distributions (e.g., the sec-

ond and the fifth dimensions of the identity code show two noticeable peaks), indicating potential entanglement in those dimensions. In contrast, the distributions produced by our method are more Gaussian-like, which supports better disentanglement and enables smoother interpolation in the latent space. Table 4.8 reports the mean KL divergence for each method. We observe that our method consistently achieves the lowest KL divergence across all code types.

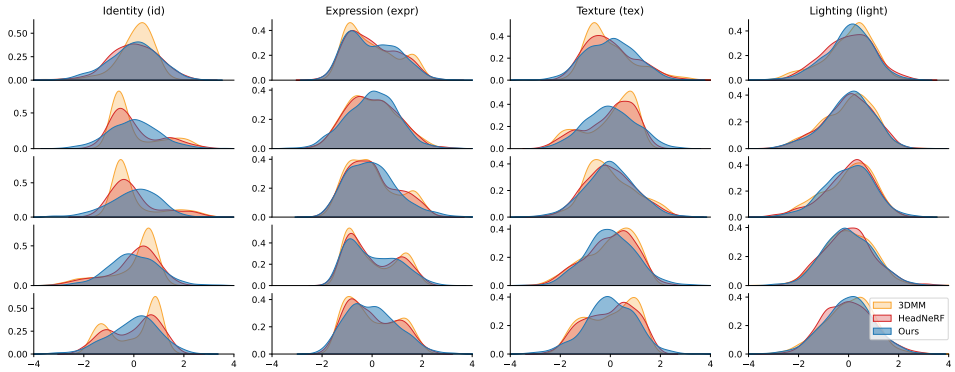


Figure 4.10: Visualization of the probability distributions of the first five dimensions for each code group (identity, expression, texture, and lighting) from 3DMM, HeadNeRF, and our method. All values are z-score normalized, so the x-axis represents the number of standard deviations from the mean. Each row corresponds to one dimension, ordered from top to bottom.

Table 4.8: Code space normality evaluation results (mean KL divergence from standard normal distribution ↓).

Method	Identity (id)	Expression (exp)	Texture (tex)	Lighting (light)
3DMM	0.2016	0.2148	0.1871	0.0688
HeadNeRF	0.0993	0.1684	0.1005	0.0565
Ours	**0.0557**	**0.0961**	**0.0526**	**0.0548**

Ablation Study

We conduct the ablation study to gain insights into the components of our design's impact on the model's overall performance. We organize our ablation study into two experiments. The first experiment studies the impact of loss terms in our train-

ing loss function. The second experiment investigates the significance of utilizing a face encoder to learn the style code space.

In the first experiment, we study the contribution of our loss terms to rendered image quality. In the reconstruction loss, we apply the photometric loss term $\mathcal{L}_{photo}$ and perceptual loss term $\mathcal{L}_{perc}$. The reconstruction loss is utilized in both stages of our training. In the second stage, we used adversarial learning. The adversarial learning alternately optimizes G and D, with corresponding loss terms. For convenience, we define adversarial learning loss $\mathcal{L}_{adv}$ as both $\mathcal{L}_G$ and $\mathcal{L}_D$. Figure 4.11 shows the example reconstruction results using our model trained with different loss combinations. Since $\mathcal{L}_{photo}$ is the most commonly used loss term in face reconstruction works, we test training our model solely on $\mathcal{L}_{photo}$, as shown in Figure 4.11 (A). It can be observed that when we only use $\mathcal{L}_{photo}$, the reconstructed result appears very blurry, and the identity does not match the target image well. When we use both $\mathcal{L}_{photo}$ and $\mathcal{L}_{perc}$ in training our model, as shown in Figure 4.11 (A, B), the reconstructed face appears closer to the target image. However, the reconstruction is still blurry. We attribute the blurry results to two main factors. First, both $\mathcal{L}_{photo}$ and $\mathcal{L}_{perc}$ primarily capture low-frequency and mid-frequency features, which often leads to overly smooth reconstructions that lack fine-grained details [15, 101]. Second, the potential inaccuracy of camera parameters in the context of in-the-wild images further contributes to the blur. Since the original camera intrinsics are unavailable, we follow standard practice by using fixed intrinsic parameters. As a result, the estimated extrinsic parameters (camera poses) can be imprecise. This leads to slight misalignment between the projected 3D representations and the 2D images, which in turn causes a loss of sharpness in the reconstructed outputs.

When we add the adversarial learning loss, the reconstruction quality improves significantly, as shown in Figure 4.11 (A, B, C). We also test the three models on the FFHQ dataset; the results are shown in Table 4.9. Quantitative results are consistent with the visual quality, suggesting that all the above-mentioned loss terms are important in our model training.

As both reconstruction loss terms and inaccurate camera poses can contribute to image blurriness, we conduct an additional experiment to isolate the effect of pose inaccuracy on reconstruction quality. We train two groups of models: one

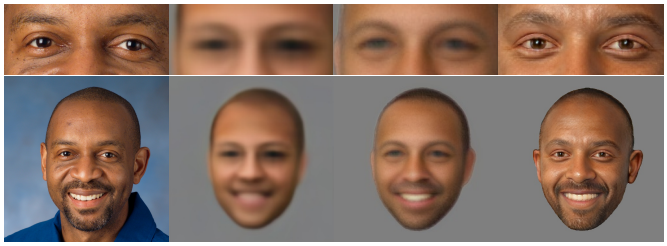


Figure 4.11: Ablation study on the loss function. **A**: $\mathcal{L}_{photo}$; **B**: $\mathcal{L}_{perc}$; **C**: adversarial loss $\mathcal{L}_{adv}$.

Table 4.9: Ablation study on the loss terms.

Loss Terms			**Evaluation Metrics**			
$\mathcal{L}_{photo}$	$\mathcal{L}_{perc}$	$\mathcal{L}_{adv}$	**L1** $\downarrow$	**LPIPS** $\downarrow$	**SSIM** $\uparrow$	**PSNR** $\uparrow$
✓	✗	✗	0.322	0.494	0.731	15.341
✓	✓	✗	0.281	0.409	0.681	15.824
✓	✓	✓	**0.126**	**0.257**	**0.814**	**22.269**

with the complete training objective including adversarial loss (w/ Adversarial Loss), and one without it (w/o Adversarial Loss). In both groups, we introduce synthetic noise to the yaw and pitch angles of the estimated camera poses, using two levels of noise strength: $\pm 0.5^\circ$ and $\pm 1.0^\circ$, uniformly sampled within those ranges. As shown in Figure 4.12, reconstruction quality consistently degrades as pose noise increases, regardless of the use of adversarial loss. The result supports our hypothesis that inaccurate poses contribute to image blurriness. Additionally, we observe a consistent quality gap between models trained with and without adversarial loss, which we attribute to the adversarial objective encouraging sharper and more visually realistic outputs, as discussed previously.

A qualitative example is shown in Figure 4.13. As camera pose accuracy decreases, the reconstructed image quality visibly degrades. The model trained with adversarial loss tends to produce more noticeable artifacts under higher pose noise, likely due to its sensitivity to misalignment. However, it still preserves better local sharpness compared to the model trained without adversarial loss.

We do not conduct ablation studies on $\mathcal{L}_{code}$ and $\mathcal{L}_{seg}$ because $\mathcal{L}_{code}$ plays a

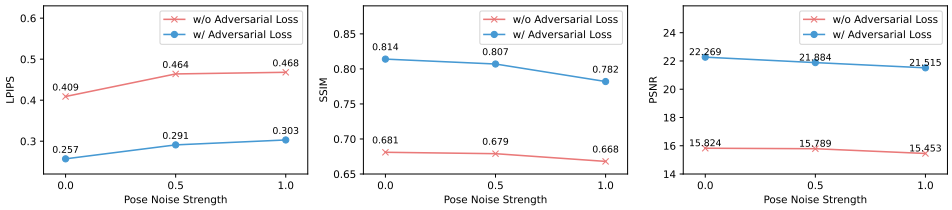


Figure 4.12: Image reconstruction quality metrics (LPIPS↓, SSIM↑, PSNR↑) versus varying camera pose noise strengths. The x-axis indicates the magnitude of synthetic camera pose noise in degrees.

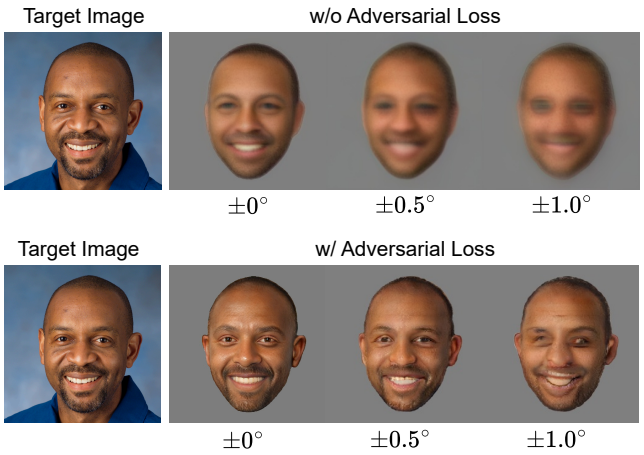


Figure 4.13: Qualitative comparison of reconstructed face images from models trained with varying levels of synthetic camera pose noise.

crucial role in learning the disentangled latent space, while $\mathcal{L}_{seg}$ is specifically designed to enable the network to simultaneously generate a segmentation mask that separates the rendered head from the background. Removing either $\mathcal{L}_{code}$ or $\mathcal{L}_{seg}$ is not expected to degrade the rendering quality, as they primarily support disentanglement and post-processing, rather than directly influencing image fidelity.

In our second experiment, we evaluate the impact of different input code configurations for the generator. We design three ablation groups. The first two groups bypass the face encoder (i.e., no Stage 1 training) and instead use: (1) the estimated 3DMM codes directly, and (2) the HeadNeRF-optimized 3DMM codes (3DMM + offset). The third group follows the same setup as our complete model, but with

one key difference: the face encoder is trained from scratch without initialization from pretrained ArcFace weights. We evaluate the reconstruction performance on our FFHQ test set; the results are shown in Table 4.10.

Table 4.10: Ablation study on the input style code.

Input Code	**L1** ↓	**LPIPS** ↓	**SSIM** ↑	**PSNR** ↑
3DMM	0.407	0.466	0.612	13.074
3DMM + Offset	0.282	0.406	0.675	15.771
Ours (w/o ArcFace)	0.261	0.386	0.609	16.219
Ours	**0.126**	**0.257**	**0.814**	**22.269**

Some example reconstructions from this experiment are shown in Figure 4.14. When using the fitted 3DMM code as input, the reconstructed faces exhibit inaccurate hairstyles due to the limited expressiveness of the 3DMM code space. Using HeadNeRF's improved 3DMM code (3DMM + offset) yields better reconstruction quality, as the offset enables the representation to capture additional information, including approximate hairstyle and more accurate facial geometry. However, fixed input codes still result in noticeable artifacts in the generated images.

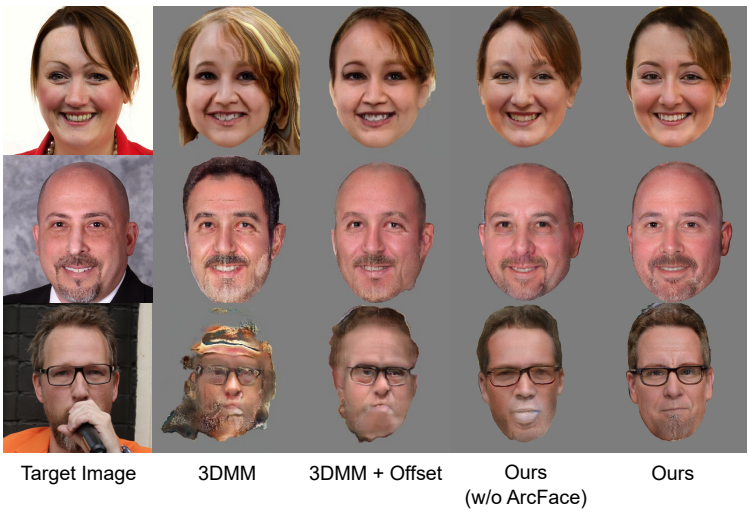


Figure 4.14: Ablation study on the input codes.

We also observe that training the face encoder from scratch (i.e., w/o ArcFace initialization) leads to lower reconstruction accuracy and reduced image quality. To

clarify the performance gap, our training dataset lacks identity labels. It contains significantly fewer images (approximately 66,000 unlabeled images from FFHQ) compared to ArcFace's training dataset, which comprises 11.96 million images across 493,000 labeled identities. Moreover, ArcFace leverages identity labels through its Additive Angular Margin Loss, enabling it to capture robust discrimination among varying facial identities. These factors limit FFHQ's effectiveness in learning robust and representative features in the latent space.

In a more challenging example (the third row in Figure 4.14), where the face is partially occluded by a hand, our full method is still able to reconstruct the occluded region reasonably well, demonstrating the benefit of our learned representation.

4.4.4 Applications

We implemented multiple applications to showcase the potential of StyleMorpheus further.

Style-Mixing

Inspired by StyleGAN-based image generators [64, 87], we utilize style mixing to demonstrate our model's disentangled control capabilities. In our case, we mix the z style codes. We define source image I_{src} and target image I'_{tgt}. Their fitted style codes are $z_{src} = \{z_{id}, z_{expr}, z_{tex}, z_{light}\}$ for source image I_{src} and $z'_{tgt} = \{z'_{id}, z'_{expr}, z'_{tex}, z'_{light}\}$ for target image I'_{tgt}. We replace one segment of the source code with its corresponding target code to get the mixed code z^*. For example, if we want to use the target to provide the facial expression (expr.), then we replace z_{expr} with z'_{expr}: $z^* = \{z_{id}, z'_{expr}, z_{tex}, z_{light}\}$. Figure 4.15 shows the style mixing results.

Disentangled Face Morphing

We further demonstrate our method's ability to achieve disentangled control using a face morphing example, as illustrated in Figure 4.16. In this experiment, we compare our approach with HeadNeRF. We select two subjects, A and B, reconstruct their latent codes, and then perform linear interpolation between these codes to morph from subject A to subject B. Specifically, we interpolate one of the four

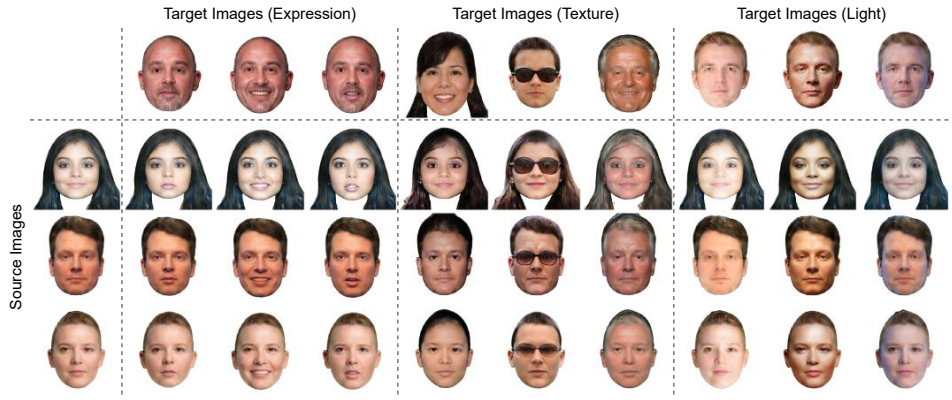


Figure 4.15: Disentangled style mixing of latent space. The leftmost column shows source images and the first row shows the target images. We divide the target images into three groups: expression, texture, and light. The target image provides the corresponding semantic group of code, and the source image provides the rest semantic groups of code.

style attributes (identity, expression, texture, and lighting) at a time, while keeping the remaining attributes fixed to those of subject A. As depicted in the figure, our approach consistently produces higher-quality renderings compared to HeadNeRF, while effectively preserving a clear and satisfactory disentanglement among the style attributes. Additionally, we observe that the identity code primarily controls identity-related features, such as face shape, hairstyle, and age, whereas the texture code controls attributes like hair and skin color.

Facial Part Color Editing

We propose a novel approach for editing the color of facial parts such as hair and skin, using StyleMorpheus. Starting with the face code $z = \{z_{id}, z_{expr}, z_{tex}, z_{light}\}$ and a camera pose p, we render the facial image $I = G(z, p)$. To edit the part color, we first run the face parsing model on I, to get the corresponding part's binary mask $\mathbf{M}_{part}$. Then, we define the texture code offsets Δz_{tex} (initialized as zeros), to form the edited face code $z' = \{z_{id}, z_{expr}, z_{tex} + \Delta z_{tex}, z_{light}\}$. We render the edited image as $I' = G(z', p)$. We define an image I^* that has the same size as I and I', and has pure color (all pixels have the same intensity) as the target part

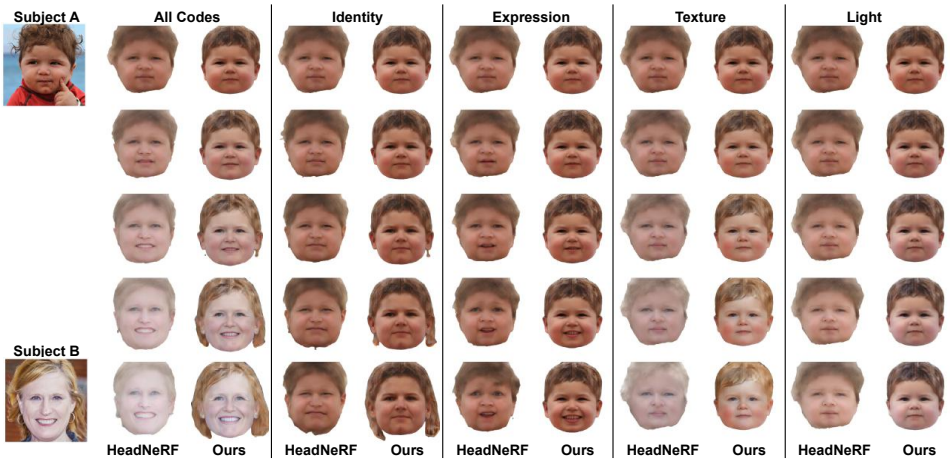


Figure 4.16: Face morphing results demonstrating linear interpolation between reconstructed latent codes of subjects A and B.



Figure 4.17: Part color editing examples. In the first row, we edit the hair color; in the second row, we edit the skin tone. The average color of the corresponding part is shown in the top right patch.

color. Our objective is then formulated as:

$$\underset{\Delta z_{tex}}{argmin}\ \mathcal{L}_{color} + \mathcal{L}_{rest} + \|\Delta z_{tex}\|_2^2, \tag{4.19}$$

where

$$\mathcal{L}_{color} = \frac{\|I' \odot \mathbf{M}_{part} - I^* \odot \mathbf{M}_{part}\|_2^2}{\|\mathbf{M}_{part}\|_1}, \tag{4.20}$$

$$\mathcal{L}_{rest} = \frac{\|I' \odot (1 - \mathbf{M}_{part}) - I \odot (1 - \mathbf{M}_{part})\|_2^2}{\|1 - \mathbf{M}_{part}\|_1}, \tag{4.21}$$

and $\|\Delta z_{tex}\|_2^2$ is a regularization term. The purpose of $\mathcal{L}_{color}$ is to allow the average part color to approach the target part color. Both $\mathcal{L}_{rest}$ and $\|\Delta z_{tex}\|_2^2$ help to regulate the code offset from being too large and to ensure the edited face still resembles the original face. Figure 4.17 shows our part color editing results. Our face part color editing feature enables users to further customize a reconstructed face.

Real-Time Rendering and Virtual Reality Demonstration

Real-time performance is crucial in virtual reality applications, as it provides seamless interactions and user immersion, thereby enhancing the overall experience and preventing discomfort. Olano et al. suggest that the frames-per-second (FPS) rate should be at least 15 to be considered real-time rendering [133]. We test several 3D-aware face models on our devices to benchmark their ability to render in real-time. For a fair comparison, we set the rendered image size to 512×512 and the number of points sampled per ray to 64. We benchmark on two devices. The first device is a personal computer with an Intel i7-11700F CPU, 32GB RAM, and a mid-range NVIDIA RTX 3070 (8GB VRAM) GPU. The second device is a server computer equipped with an AMD Threadripper PRO 5995WX CPU, 512GB of RAM, and an NVIDIA RTX A6000 (48GB VRAM) GPU. The benchmark results are shown in Table 4.11. As we can see, our method has achieved satisfactory real-time performance even on a personal computer. Methods such as StyleNeRF [64], MoFaNeRF [247], and Next3D [173] fail to meet our real-time benchmark on mid-range GPUs.

We developed a simple virtual reality application for demonstrating the proposed StyleMorpheus in a real-time 3D-aware image synthesis setting. As illustrated in the left side of Figure 4.18, our setup comprises a high-resolution computer display, a webcam, and a person as the viewer. We use the webcam to capture the viewer's face and estimate its pose (including its relative location to the center of the screen). This estimated pose is then used to determine the camera's perspective for rendering the face. As the viewer's pose changes, the rendered face pose

Table 4.11: Frames-per-second ↑ of different 3D-aware face models on different devices.

Method	**RTX 3070** (8GB VRAM)	**RTX A6000** (48GB VRAM)
StyleNeRF [64]	5	8
MoFaNeRF [247]	Out of memory	2
Next3D [173]	11	19
HeadNeRF [71]	24	**56**
Ours	**26**	53

The red color indicates that real-time has not been reached.

and location adjust accordingly. This creates the illusion of peering through a window to perceive a genuine 3D face in physical space. The right side of Figure 4.18 depicts some examples of the rendered face on the display. This demonstration shows the potential of our StyleMorpheus in immersive virtual reality applications, such as telepresence.

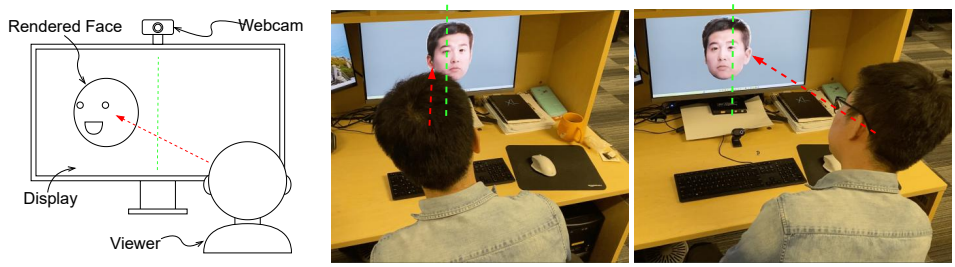


Figure 4.18: Our virtual reality demonstration. The left side shows the setup of our demonstration. It includes a high-resolution display, a calibrated webcam, and a viewer. The viewer sits in front of the display. When the viewer's head moves, we render the face using the corresponding camera pose to create the illusion of a 3D effect. The right side shows the rendered faces viewed from different directions. The green dashed line indicates the horizontal center of the screen. The red dashed line with an arrow indicates the viewer's viewing direction.

4.4.5 Limitations and Future Work

Through our experiment, we identified some limitations with our current model. In rare cases of single-image face reconstruction, when the face pose is large and the camera parameters are not accurate, the reconstruction tends to be inaccurate. This limitation arises because in-the-wild images typically lack ground-truth camera intrinsics. Following standard practice, we use fixed intrinsic parameters, which leads to a mismatch in perspective distortion between the rendered face and the target image. This discrepancy becomes especially noticeable when the face is viewed from different angles (e.g., a frontal view), where distortion artifacts can appear, as shown in Figure 4.19.



Figure 4.19: An example of the failure case. When the estimated camera pose is not accurate enough, the reconstructed face gets distorted in other viewing directions.

In future work, we plan to explore the self-supervised optimization of camera parameters during the reconstruction training process. This could lead to improved 3D-to-2D alignment accuracy, thereby enhancing the quality of the learned 3D face reconstruction. We also plan to explore integrating learnable pose estimation to our framework to reduce our reliance on off-the-shelf pose estimation tools.

Another limitation is the exaggerated facial expressions. Since the in-the-wild images we use do not cover a wide range of exaggerated facial expressions, our model often fails to reconstruct face images with exaggerated facial expressions. To tackle this challenge, we plan to leverage in-the-wild video face datasets, such as VFHQ [212], to improve the representation ability of facial expressions.

4.4.6 Summary

We introduce StyleMorpheus, the first StyleGAN-based, 3D-aware morphable face model. StyleMorpheus is trained only on in-the-wild data and enables the recon-

struction of high-fidelity faces with editable attributes from single-view images. Our approach utilizes a face autoencoder to learn a disentangled facial representation, enabling the independent manipulation of identity, expression, texture, and lighting. We design a StyleGAN-based NeRF decoder that renders facial images from various viewpoints, conditioned on the encoded face representation in the disentangled style code space. Through adversarial learning, StyleMorpheus excels at capturing intricate facial details, resulting in photorealistic reconstructions. It outperforms existing 3D-aware morphable face models in reconstruction and rendering quality, and achieves independent control over key facial attributes due to its disentangled style code space. We also demonstrate several practical applications to showcase the real-world utility of StyleMorpheus.

4.5 Chapter Summary

In this chapter, we explore achieving photorealistic quality while reducing cost by using in-the-wild images as training data for 3D head creation. To this end, we leverage the 3D-aware implicit representation method, Neural Radiance Field (NeRF), to represent the 3D head. However, due to the lack of explicit 3D geometry, 3D-aware models are typically less controllable in terms of shape deformation and appearance editing. Therefore, we follow the idea of the parametric 3D face model (3DMM) to construct a parametrically controllable 3D-aware morphable face model. We first study an existing 3D-aware parametric face model, HeadNeRF [71], and propose a learning-based single-image face reconstruction framework that enhances the reconstruction quality of HeadNeRF. However, HeadNeRF is limited due to its reliance on multi-view data for training. The first issue is that collecting multi-view images requires a lab setup and is costly. The second issue is that such datasets often lack diversity due to the limited number of participants. To address these issues, we propose a new 3D-aware morphable face model, StyleMorpheus. StyleMorpheus is based on NeRF and StyleGAN to achieve both 3D consistency and improved generalization ability. Different from existing StyleGAN-based NeRF methods, we design the network architecture to achieve parametric controllability. In particular, we utilize four groups of style codes, namely identity, expression, appearance, and lighting, rather than a sin-

gle style code as in the original StyleGAN, enabling disentangled parametric control similar to 3DMM. The proposed StyleMorpheus achieves better photorealistic reconstruction quality than existing 3D-aware parametric face models, including HeadNeRF, while also providing improved controllability.

Chapter 5

Photorealistic 3D Head Avatar Creation

The Matrix isn't real. — Trinity, The Matrix (1999)

In Chapter 3, we studied extending the 3D mesh-based parametric morphable face model (3DMM) with intuitive editing and shape stylization capabilities. However, the 3D mesh-based representation has limitations in learning from 2D image data and is less flexible for representing complex geometries. In Chapter 4, we investigated using the neural implicit 3D representation method, Neural Radiance Field (NeRF), to enable training from in-the-wild 2D images while preserving the parametric controllability of 3DMM. Despite the strong generalization ability of our NeRF-based method, it cannot be further personalized when only a short video of a person's head is available. In addition, NeRF-based 3D head methods remain computationally expensive due to the need to sample a large number of points during rendering. In this chapter, we build on the emerging 3D representation method, 3D Gaussian Splatting (3DGS) [90], to achieve real-time animatable head avatar construction using only short videos.

5.1 Introduction

The creation of photorealistic and controllable 3D head avatars has become a popular topic due to their employment in video gaming, VR and AR, filmmaking, telepresence, and many other fields. Three important factors must be considered when creating a 3D head avatar: **efficiency**, **quality**, and **controllability**. Efficiency is related to both the training and the rendering processes. Quality refers to achieving photorealistic results that resemble real human appearances. Controllability involves the ability to manipulate facial expressions easily, head poses, and camera views. Meeting these requirements ensures that the avatars can be effortlessly integrated into various applications. While existing mesh-based and NeRF-based methods may address some of these factors, they often fail to address all of them. 3DGS is a hybrid 3D representation that provides explicit 3D information, such as the locations and scales of 3D Gaussian points, while also representing a radiance field through these discrete points to produce high-quality renderings. The core idea of 3DGS is to utilize 3D geometry primitives, specifically 3D Gaussians, to represent 3D scenes, enabling smooth and efficient rendering of complex scenes by optimizing the density and attributes of the 3D Gaussians. It offers a promising way to model 3D head avatars, providing more flexibility than mesh-based methods and also more efficient rendering than NeRF-based methods.

Despite these advantages, the creation of controllable 3DGS-based head avatars remains time-intensive, often requiring tens of minutes to hours. An important step of learning a 3DGS-based head avatar is the initialization of 3D Gaussians. Some works randomly initialize the locations of 3D Gaussians [29, 196], while most initialize the locations based on the 3D head mesh to make early-stage learning more efficient [41, 61, 98, 110, 118, 120, 145, 153, 159, 210, 215]. However, a 3D Gaussian contains more information beyond its location, and only initializing the locations of the 3D Gaussians still leads to slow convergence. We assume that a single-image-based reconstruction model can provide a good initialization of the 3D Gaussians. Nevertheless, before this work, existing 3DGS-based animatable head avatar creation methods do not support the use of a single 2D image. To address this, we propose the **Déjà-vu** method (Section 5.3), which first obtains a generalized model of the head avatar and then personalizes the result. The general-

ized model is trained on large 2D (synthetic and real) image datasets and provides a well-initialized 3D Gaussian head that is further refined using a monocular video to achieve the personalized head avatar. For personalizing, we propose learnable expression-aware rectification blendmaps to correct the initial 3D Gaussians, ensuring rapid convergence without the reliance on neural networks. Déjà-vu is the first approach capable of reconstructing a 3DGS-based head model from a single input image, with personalization enabled through refinement from a monocular video.

While 3DGS-based methods achieve a higher degree of photorealism than traditional mesh-based methods, they are more computationally intensive and struggle to maintain real-time performance when multiple avatars are rendered simultaneously [108, 145, 159, 210, 228]. In computer graphics, level of detail (LOD) techniques for 3D meshes are widely used to balance visual quality and efficiency [72]. However, applying LOD to 3DGS is challenging due to the unstructured nature of Gaussian point clouds and the need for dynamic animation in head avatars. Conventional LOD methods provide only a few discrete levels (typically 3 to 4 levels) [42, 156], which can cause unsmooth visual effects when switching between levels. To address this gap, we propose the **ArchitectHead** method (Section 5.4), the first framework for creating 3D Gaussian head avatars that support continuous control over LOD. Our key idea is to parameterize the Gaussians in a 2D UV feature space and propose a UV feature field composed of multi-level learnable feature maps to encode their latent features. A lightweight neural network-based decoder then transforms these latent features into 3D Gaussian attributes for rendering. ArchitectHead controls the number of Gaussians by dynamically resampling feature maps from the UV feature field at the desired resolutions, enabling efficient and continuous control of LOD without retraining.

Beyond reconstructing and rendering 3D head avatars, 3DGS-based head models can also serve as a powerful tool for synthetic data generation. As an application, we explore using 3DGS-based head models to generate synthetic training data for an image-based 3D face reconstruction task. Existing image-based 3D face reconstruction methods rely on a virtual camera to project the reconstructed 3D face onto the 2D image plane for comparison with the input image, a crucial step for accurate results. To simplify the reconstruction process, these meth-

ods often use fixed camera intrinsics and assume minimal perspective distortion [38, 48, 235, 248], overlooking the varying distortion levels in in-the-wild images and leading to inaccuracies in reconstructed 3D face shapes. The camera field-of-view (FOV) represents the extent of the scene visible to the camera, determined by the focal length and sensor size. In real-world scenarios, images are captured with varying focal lengths and distances. For instance, selfies often use cameras with a small focal length and close proximity to the face, resulting in a large FOV and significant perspective distortion [164], with which existing reconstruction methods that assume a small virtual camera FOV struggle [85]. However, estimating an appropriate virtual camera FOV remains challenging due to the lack of publicly available datasets containing images of the same faces captured under varying FOVs. To address this, we propose to estimate the virtual camera's optimal FOV for a given image, enabling consistent 3D face reconstruction across varying distortion levels (Section 5.5). We introduce two synthetic datasets, PanoHead-FOV and NeRSemble-FOV, developed using state-of-the-art 3D face modeling methods [3, 219] and publicly available face datasets [87, 93], where the latter benefits from a 3DGS-based head model. The two datasets are used to train and evaluate an FOV estimation network (**FOV-Net**) for improving 3D face reconstruction in challenging cases with extreme perspective distortion.

Together, these three methods advance 3DGS-based head avatar creation along three complementary directions: enabling fast, single-image-initialized personalization (Déjà-vu); supporting on-the-fly level-of-detail control (ArchitectHead); and demonstrating the use of 3DGS-based head models for synthetic data generation in 3D face reconstruction (FOV-Net).

5.2 Related Works

5.2.1 NeRF-Based 3D Face and Head Models

The original Neural Radiance Field (NeRF) method employs a neural network to represent the static 3D scene learned from multi-view images [128]. NeRF-based 3D head methods support learning from images and videos and can achieve photorealistic results, but are computationally intensive due to the need for extensive

point sampling per frame and are less accurate in terms of geometry [173, 182, 195]. MoFaNeRF adapts NeRF to model the human face with parametric control by incorporating facial codes akin to 3DMM coefficients as additional network inputs [247]. HeadNeRF enhances rendering speed through a coarse-to-fine strategy, utilizing NeRF to render low-resolution feature maps and a 2D neural rendering network for producing high-resolution head images [71]. 3DMM-RF leverages 3DMM-based synthetic face images for training, achieving more accurate fitting over the facial region [52]. Inspired by the success of 2D StyleGANs [88, 89] in generating photorealistic face images, StyleNeRF integrates a style-based network architecture into NeRF and trains the network through adversarial learning as in StyleGAN [65]. EG3D further refines the network architecture by forming a tri-plane representation for volume rendering of the raw image [25]. Due to efficiency considerations, the tri-plane generally has low resolution, prompting the use of a 2D super-resolution module to enhance the final image resolution. This tri-plane NeRF scheme is now widely adopted by numerous 3D-aware head models [3, 32, 83, 121, 173, 214]. Although NeRF-based StyleGAN methods generate photorealistic rendered head images, they can introduce artifacts, such as noticeable flickering during animations. Moreover, these models do not support the customization of the reconstructed head based on a person's video data, limiting their applicability in more personalized head avatar scenarios. In contrast, the Déjà-vu framework we propose here produces an explicit 3D Gaussian-based head avatar that can be rendered directly through 3DGS [90] and optimized using video data to enhance personalization.

5.2.2 3D Gaussian-Based Head Avatars

Like the original NeRF, the 3D Gaussian Splatting (3DGS) is designed to represent and render static 3D scenes [90]. 3DGS has emerged as an efficient and photorealistic representation for 3D head avatars, capable of real-time rendering, and can be considered a point-based representation that explicitly represents the head as a set of 3D points [29, 108, 145, 239]. To achieve controllability, 3DGS-based head avatar methods typically employ 3DMM. This can be done by rigging the Gaussians to mesh triangles [108, 145, 159, 238] or initializing them via UV

maps [107, 169, 210, 219]. GaussianAvatars introduces a technique to rig the 3D Gaussians to the FLAME model, where the Gaussians are anchored to the triangle facets of the mesh [145]. Similarly, SplattingAvatar proposes a method to embed 3D Gaussians into any head mesh model [159]. SplatFace aligns the 3D Gaussians with the underlying head mesh through a non-rigid process and jointly optimizes both the Gaussians and the mesh [118].

Numerous 3DGS head avatar methods leverage neural rendering to achieve photorealism by incorporating high-dimensional latent features into each 3D Gaussian and utilizing neural networks to decode these features to produce the final images [9, 41, 61, 98, 110, 153, 196, 198, 215]. For instance, GaussianHead and Gaussian Splatting Decoder employ a tri-plane representation to learn the 3D Gaussian latent features [9, 196]. HeadGaS utilizes a Multi-layer Perceptron (MLP) network to decode the latent features of each 3D Gaussian into color and alpha blending weights [41]. Additionally, NPGA and Gaussian Head Avatar initially render a 2D feature map using a modified 3DGS, which is then decoded into an image through 2D neural rendering [61, 215]. However, these neural-rendering-based methods deviate from the simplicity of traditional 3DGS, resulting in reduced compatibility with existing 3D engines. In addition, some methods employ a "canonical + deformation" strategy which leverages a neural network, typically an MLP, to estimate the deformation for each 3D Gaussian [29, 110, 210]. Approaches like HeadGaS and 3D Gaussian Blendshapes integrate the concept of blendshapes by learning a set of 3D Gaussians as expression bases [41, 120, 219].

Another research direction focuses on generalization, using large-scale prior models for few-shot or single-image personalization [68, 94, 107, 238]. Diffusion-based frameworks have also been explored to improve monocular video supervision and robustness to novel views [58, 179, 180].

Despite the success of existing 3DGS-based head avatar methods, this work takes us one step further, introducing the first single-image-based reconstruction of a 3D Gaussian head avatar, which provides robust initialization for our video-based optimization. Our 3D Gaussian head avatar aligns with the original 3DGS and does not require a neural network for animation, ensuring compatibility and ease of integration with standard 3D engines.

5.3 Method 1: Creating Controllable 3D Gaussian Head-Avatars with Enhanced Generalization and Personalization Abilities (Déjà-vu)

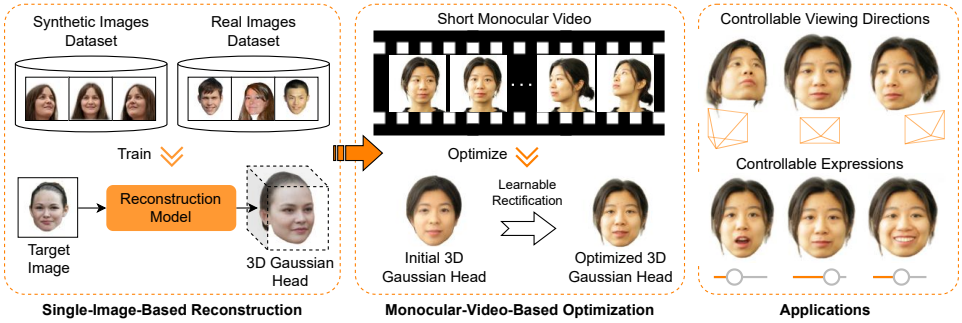


Figure 5.1: The proposed Déjà-vu framework first trains a reconstruction model on large face image datasets and serves as a generalized base. This model initializes the 3D Gaussian head, which is then optimized to personalize the avatar to match the person in the video.

5.3.1 Overview

Recent advancements in 3D Gaussian Splatting (3DGS) have unlocked significant potential for modeling 3D head avatars, providing greater flexibility than mesh-based methods and more efficient rendering compared to NeRF-based approaches. Despite these advancements, the creation of controllable 3DGS-based head avatars remains time-intensive, often requiring tens of minutes to hours. Three important factors must be considered when creating a 3D head avatar: **efficiency**, **quality**, and **controllability**. Efficiency is related to both the training and the rendering processes. Quality refers to achieving photorealistic results that resemble real human appearances. Controllability involves the ability to manipulate facial expressions easily, head poses, and camera views. While existing methods may address some of these factors, they often fail to address all of them. In particular, most existing 3DGS-based animatable head avatar methods take tens of minutes or even hours to train an avatar for a single person, significantly limiting their wide application. We assume that a single-image-based reconstruction model can provide a good ini-

tialization of the 3D Gaussians; however, before this work, existing 3DGS-based animatable head avatar creation methods do not support the use of a single 2D image. Table 5.1 summarizes the features supported by different 3D head avatar creation methods.

Table 5.1: Comparison of supported features.

	Mesh-based Methods	**NeRF-based Methods**	**Existing 3DGS Methods**	**Ours**
Single-Image Reconstruction	✓	✓	✗	✓
Easy Animation	✓	✗	✓	✓
Flexible Shape	✗	✓	✓	✓
Fast Training on Videos	N/A	✗	✗	✓
Efficient Rendering	✓	✗	✓	✓

To address this gap, we introduce the "Déjà-vu" framework, which enables efficient creation of high-quality and controllable 3D Gaussian head avatars by first obtaining a generalized model of the head avatar and then personalizing the result. The generalized model is trained on large 2D (synthetic and real) image datasets and provides a well-initialized 3D Gaussian head that is further refined using a monocular video to achieve the personalized head avatar. Specifically, we parameterize the 3D Gaussians into a specialized UV map representation (a 2D representation of a 3D object's surface), which we call the UV Gaussian map, aligned with the mesh-based FLAME model [109] to enable shape initialization and controllable animation. We train a reconstruction network to generate UV offsets that will be used to correct the FLAME-initialized 3D Gaussians [210] in the context of the single-image-based reconstruction task. The training primarily uses a synthetic dataset with a large number of identities, each captured from multiple viewpoints. We propose a training scheme that randomly pairs the input and target views of the same identity (person), along with a corresponding multi-view consistency regularization loss to enhance the 3D consistency. To further improve the network's generalization ability, we also fine-tune it on real and in-the-wild images. We only need to train the reconstruction network once.

For personalizing, we use our trained single-image-based reconstruction network to provide an initial 3D Gaussian head and then propose a rectification approach to enable further optimization of the head using a monocular video of

the intended person. This rectification approach eliminates the need for a neural network by introducing learnable expression-aware UV rectification blendmaps (blendmaps), which are additional offsets to the initial 3D Gaussians. Unlike 3D Gaussian blendshapes methods [41, 120], our proposed blendmaps are designed solely to rectify the initial 3D Gaussians. Therefore, the optimization is faster than training a set of blendshapes from scratch. Experiments demonstrate that the proposed method meets its objectives. Due to our efficient design, Déjà-vu achieves a real-time animation speed of 220 frames per second (fps) at a resolution of 512×512 and a faster training speed than existing methods, reducing the training time to at least a quarter of the original. It outperforms state-of-the-art 3D Gaussian head avatars in terms of photorealistic quality, while also reducing training time consumption to at least a quarter of that of existing methods, allowing for the avatar to be produced in minutes.

In summary, our contributions are as follows:

1. We propose Déjà-vu, a framework that first develops a single-image-based 3D Gaussian head reconstruction model to initialize a 3D Gaussian head and then further optimizes the head using monocular videos to make it appear more realistic and personalized. This approach significantly reduces the training time on videos and produces state-of-the-art quality;

2. We propose expression-aware UV rectification blendmaps, which allow easy control of facial animation without the need for a neural network;

3. Our training strategy for the single-image-based reconstruction model involves the use of both synthetic and real images as training data, as well as view-consistency regularization, which enhances generalization. The reconstruction model provides a robust initialization for subsequent video-based optimization. This is the first single-image-based reconstruction model for the 3D Gaussian head.

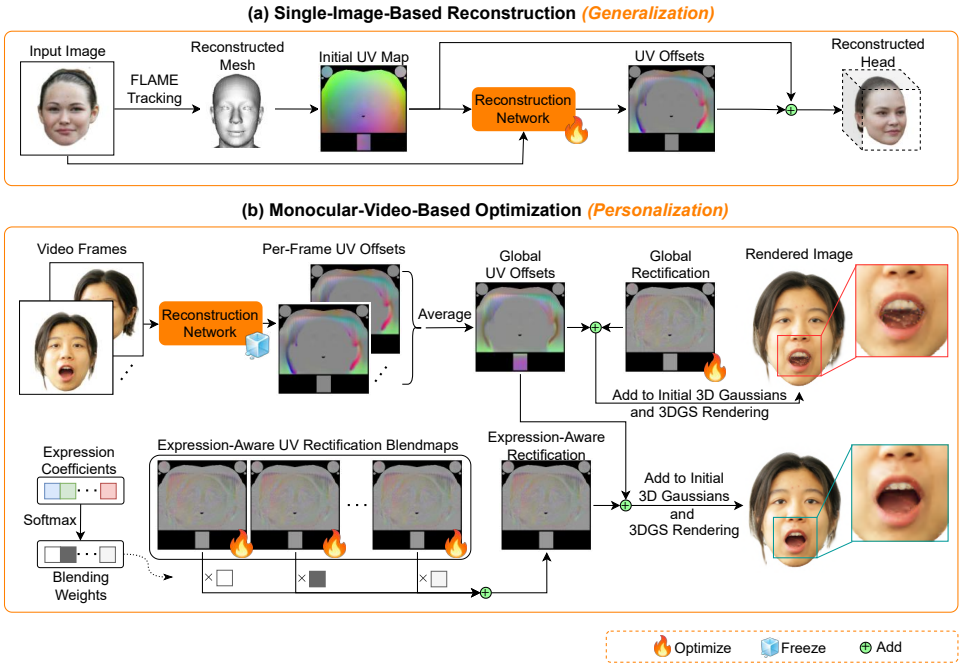


Figure 5.2: Overview of our proposed Déjà-vu framework for 3D Gaussian-based head avatar creation. (a) Single-image-based reconstruction process. (b) Monocular-video-based further optimization process.

5.3.2 The Proposed Method

UV Gaussian Map Representation

Our 3D Gaussian head avatar is based on FLAME [109] to gain controllability. We denote F as the FLAME shape model. The 3D head mesh can be constructed through: $m = F(\beta_{id}, \beta_{exp}, \beta_{jaw}, \beta_{eyes})$, where $\beta_{id} \in \mathbb{R}^{100}$ and $\beta_{exp} \in \mathbb{R}^{50}$ are the coefficients of identity and expression, respectively; $\beta_{jaw} \in \mathbb{R}^3$ and $\beta_{eyes} \in \mathbb{R}^6$ are the rotation angles for the pose of the jaw and the eyeballs, respectively. We modify the original UV correspondences of FLAME to remove the neck and add the mouth interior shape (see Figure 5.3). We develop an automatic method to generate the interior of the mouth for the FLAME mesh (see Appendix B.1 for details).

We define a set of UV maps with height and width of H and W to represent

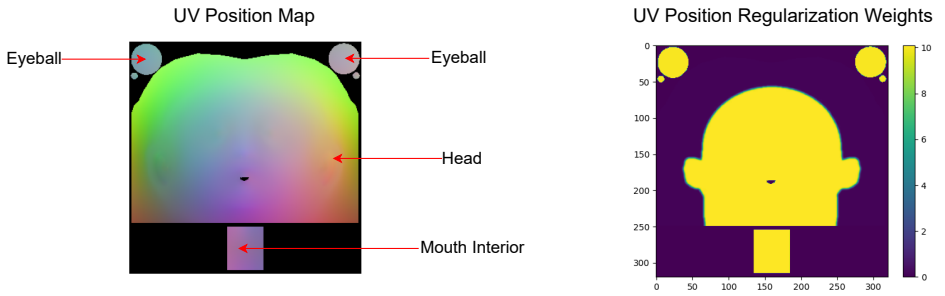


Figure 5.3: Our UV map structure (left) and UV position regularization weights (right).

the parameters of 3D Gaussians. These UV maps include U_{μ} for the 3D Gaussian positions, $U_{\mathbf{s}}$ for the scales, $U_{\mathbf{r}}$ for the rotations, U_{α} for the alpha blending weights, and $U_{\mathbf{a}}$ for the SH coefficients. Given the relatively simple surface materials of the human head, we do not consider complex anisotropic reflections and refractions. Therefore, we use SHs with a degree of zero to represent RGB colors uniformly across all directions. The number of channels in $U_{\mathbf{a}}$ is thus equal to three. Our UV Gaussian map $U \in \mathbb{R}^{H \times W \times 13}$ is created by channel-wise concatenation of the above set of UV maps. We mask out the invalid pixels (undefined locations in the UV map) in the UV Gaussian map, and each valid pixel corresponds to a 3D Gaussian. Therefore, the total number of 3D Gaussians is less than $H * W$. Figure 5.4 shows an example of a UV Gaussian map. We also illustrate the effect of scale modification in Figure 5.5. As shown, reducing the size (scales along three axes) of each 3D Gaussian makes the result appear more like a point cloud.

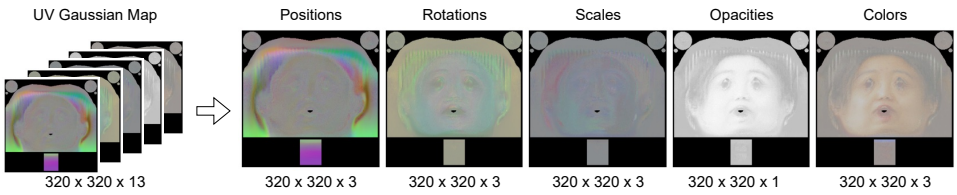


Figure 5.4: Example UV Gaussian Map. The black area indicates the invalid (unused) pixels. Both of the width and the height are set to 320.

To prepare an initial UV Gaussian map U_{init}, we rasterize the mesh shape m to



Figure 5.5: Effects of modifying the scales of 3D Gaussian points.

initialize the Gaussian positions U_μ. We empirically initialize the values of alpha weights in U_α to 0.1 and the values of scales in $U_\mathbf{s}$ to -8.3533 empirically (note that 3DGS applies exponential activation on the scales to ensure that the final scales are positive). All the other UV maps are initialized with zeros.

Single-Image-Based Reconstruction Model

Our single-image-based 3D Gaussian head reconstruction utilizes a convolutional encoder-decoder network to estimate the offsets to the initialized 3D Gaussians, as shown in Figure 5.6. The encoder takes a single image as input and generates a feature map. The decoder takes both the feature map and the initial UV position map as input and generates the final UV offsets. In each decoder block, we concatenate the feature map with the resized initial UV position map to inject the explicit shape information into the decoder.

The initial 3D Gaussian head is defined by using the initial UV Gaussian map, as detailed earlier. We implement a FLAME tracking method to derive the FLAME coefficients and the camera pose given an image. Our goal is to estimate a set of UV offsets ΔU to the initial 3D Gaussian attributes in U_{init}. The encoder processes a head image I to produce low-resolution feature maps, which are subsequently input into the decoder. Additionally, the decoder incorporates the initial UV position map U_μ to enhance the network's adaptation to the UV map layout. The corrected UV Gaussian map for the single-image-based reconstruction phase is computed as $U_{\text{SIR}} = U_{\text{init}} + \Delta U$. Using the estimated camera pose for I, we render U_{SIR} through 3DGS, with the resulting image denoted by I'_{SIR}. The reconstruction loss, which

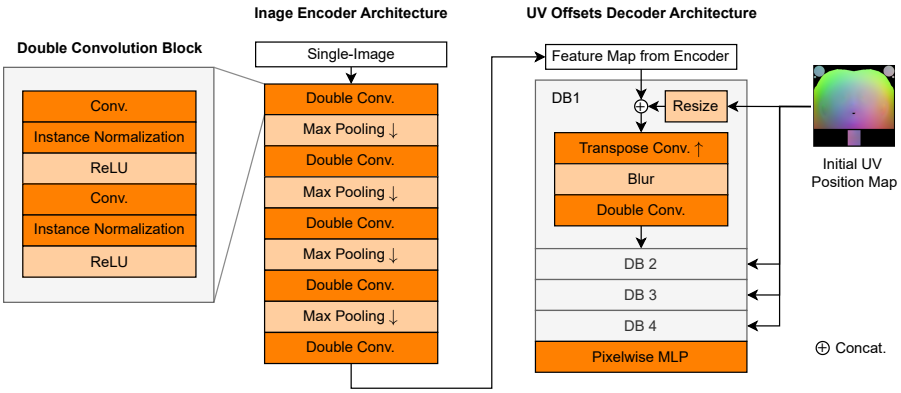


Figure 5.6: Our single-image-based reconstruction network architecture. DB stands for decoder block.

quantifies the discrepancy between I and I'_{SIR}, is computed as follows:

$$\mathcal{L}_{\text{photo}} = \lambda_{\text{RGB}}\mathcal{L}_{\text{RGB}} + \lambda_{\text{LPIPS}}\mathcal{L}_{\text{LPIPS}} + \lambda_{\text{SSIM}}\mathcal{L}_{\text{SSIM}}, \tag{5.1}$$

where $\mathcal{L}_{\text{RGB}}$ is the pixel-level L1 loss, $\mathcal{L}_{\text{LPIPS}}$ is the perceptual loss [234], $\mathcal{L}_{\text{SSIM}}$ is the differentiable SSIM loss [201], and the λs are the corresponding loss term weights.

To ensure that the positions of the 3D Gaussians do not deviate significantly from the original reconstructed FLAME shape, we introduce a position regularization term, denoted as $\mathcal{L}_{\mu}$:

$$\mathcal{L}_{\mu} = \|\Delta U_{\mu} \circ M_{\mu}\|_2, \tag{5.2}$$

where ΔU_{μ} is the network estimated UV position offsets map, $\circ$ denotes the element-wise Hadamard product operation, and M_{μ} is a predefined weights map. This map allocates more weight to the facial region and less to the scalp, ensuring that the facial area more closely conforms to the reconstructed FLAME shape, while permitting the hair, represented by the 3D Gaussians originally on the scalp, to maintain some positional variability. Similarly, we define the scale regularization term,

denoted as $\mathcal{L}_{\mathbf{s}}$ to regularize the size of the Gaussians from growing excessively:

$$\mathcal{L}_{\mathbf{s}} = \|\Delta U_{\mathbf{s}}\|_2. \tag{5.3}$$

We introduce a view consistency regularization loss, $\mathcal{L}_{\text{view}}$, to ensure stability across different views of the same identity:

$$\mathcal{L}_{\text{view}} = \frac{\sum_{\Delta U \in \mathcal{B}} \|\Delta U - \Delta \bar{U}_{\mathcal{B}}\|_2}{|\mathcal{B}|}, \tag{5.4}$$

where $\mathcal{B}$ represents a set of UV offsets from multi-view input images for the same identity, and $\Delta \bar{U}_{\mathcal{B}}$ denotes the mean UV offsets within the set. To further enhance view consistency, we also randomly pair the input and target images from the same batch of multi-view images belonging to the same identity during training. The overall loss is:

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \lambda_{\mu}\mathcal{L}_{\mu} + \lambda_{\mathbf{s}}\mathcal{L}_{\mathbf{s}} + \lambda_{\text{view}}\mathcal{L}_{\text{view}}. \tag{5.5}$$

Note that the view consistency loss is only used when the training batch is from our synthetic multi-view images. When the training batch consists of in-the-wild images, this loss term is excluded.

Figure 5.2 (a) illustrates our single-image-based reconstruction process.

Monocular-Video-Based Optimization

Using a monocular video of a person, we apply our single-image-based reconstruction to each frame and then average the estimated UV offsets to obtain the mean UV offsets for the video, denoted as $\Delta \bar{U}$. Merely using $U_{\text{init}} + \Delta \bar{U}$ to form the reconstructed 3D Gaussian head may result in a lack of personalization. To address this, we propose a two-stage optimization method designed to enhance the personalization and quality of the reconstructed 3D Gaussian head. See Figure 5.2 (b).

In the first stage (Figure 5.2 (b) top), we define the global UV rectification, ΔU_{global} as a tensor in $\mathbb{R}^{H \times W \times 13}$, which is initially set to zeros and made learnable. The resulting UV Gaussian map after applying global UV rectification is expressed

as:

$$U_{\text{global}} = U_{\text{init}} + \Delta\bar{U} + \Delta U_{\text{global}}. \tag{5.6}$$

The objective of the first stage is to optimize ΔU_{global} to minimize $\mathcal{L}$ in Equation 5.5 (excluding $\mathcal{L}_{\text{view}}$). After the first stage of optimization, the resulting 3D Gaussian head already closely resembles the person in the video. However, relying solely on ΔU_{global} can still lead to artifacts during animation, as this approach does not account for variations in complex facial expressions.

To address this limitation, we introduce a second stage (Figure 5.2 (b) bottom) that optimizes the UV rectification blendmaps $\Delta U_{\text{blend}} \in \mathbb{R}^{D\times H\times W\times 13}$, where D denotes the number of blending weights and $\Delta U^i_{\text{blend}}$ denotes the i^{th} blendmap. Our blending weights $b \in \mathbb{R}^D$ are derived from the estimated FLAME expression coefficients β_{exp}:

$$b = softmax(\beta_{exp}), \tag{5.7}$$

where b_i indicates the i^{th} blending weight. Utilizing the softmax function ensures that the blending weights are positive values with a sum of exactly 1.0. We initialize the blendmaps with the ΔU_{global} learned in the first stage. The final UV Gaussian map after rectification is expressed as:

$$U_{\text{blend}} = U_{\text{init}} + \Delta\bar{U} + \sum_{i=1}^{D} b_i \Delta U^i_{\text{blend}}. \tag{5.8}$$

We optimize ΔU_{blend} with the same objective as in the first stage.

5.3.3 Experiments

Datasets

We prepare two datasets for training our single-image-based reconstruction model. The first is a synthetic dataset created using the 3D-aware face image generation model, PanoHead [3], which synthesized images for 18,000 identities across 25 pre-defined camera views (see Figure B.3 in Appendix B.2 for an example). We utilize the DeepFace PyTorch Toolkit [157] to estimate age, gender, and emotion statistics on our synthetic dataset, as shown in Figure B.4 in Appendix B.2.

The second dataset comprises real face images from the FFHQ dataset [87] to enhance the robustness of our network on authentic images. The reason for using the FFHQ dataset is its wide coverage of faces from various ethnicities, genders, and age groups. However, the original FFHQ dataset primarily focused on the facial region; therefore, the facial region occupies a larger portion of the aligned image. This usually causes the head not to be fully visible in the image (e.g., part of the hair is outside the cropped image). We first collect the raw images (without alignment) used by FFHQ and realign the faces using our standard to cover the entire head (see Figure B.5 in Appendix B.2). We filtered out images where faces are obscured by hats or clothing, or are of low quality, resulting in 38,000 usable images. Of these, the first 1,000 images are set aside for evaluation, while the remaining 37,000 images are used for training.

For video-based optimization, we utilize the dataset from PointAvatar [239], which includes monocular video clips of three subjects [239]. For each subject, we allocate the first 90% of the frames from each video clip for training purposes, reserving the remaining 10% for evaluation.

Implementation Details

We set the resolution of our UV Gaussian map to 320×320 in our experiment. The actual number of Gaussians is 74,083. The loss term weights are set as follows: $\lambda_{\text{RGB}} = 1.0$, $\lambda_{\text{LPIPS}} = 0.5$, $\lambda_{\text{SSIM}} = 0.05$, $\lambda_{\mu} = 0.5$, $\lambda_{\mathbf{s}} = 5 \times 10^{-5}$, and $\lambda_{\text{view}} = 0.1$. The rendering resolution is 512×512.

During the training of the single-image-based reconstruction network, each training step starts with training on a random batch from the synthetic dataset at a higher learning rate. This is followed by fine-tuning on a random batch from the real dataset at a reduced learning rate, specifically one-tenth of that used for the synthetic dataset. It is important to note that the $\mathcal{L}_{\text{view}}$ term is omitted during fine-tuning on the in-the-wild dataset, as multi-view images are not available. The training batch size is set to 16.

We use Adam optimizer with an initial learning rate of 1e-4 on synthetic data, and an initial learning rate of 1e-5 on real data. We decrease the learning rate by 5% after every 10,000 training steps. We start the training with a rendering resolution

of 128 and gradually increase the resolution after every 100,000 training steps until the resolution reaches 512×512. When rendering at 128×128, we do not fine-tune on real data. Every time the rendering resolution is increased, we reset the learning rates.

For video-based optimization, the distribution of steps between the first and second stages is split into a 3:7 ratio. The batch size for each step is 8. We utilize only the first ten expression coefficients to compute the blending weights, where $D = 10$. We use Adam optimizer with an initial learning rate of 0.05 for both stages of training. After every 100 training steps, we decrease the learning rate by 10%.

All the experiments are performed on one NVIDIA A6000 GPU (48GB VRAM). For additional details, please refer to the supplementary material.

Single-Image-Based Reconstruction Results

We compare the performance of our single-image-based reconstruction model with that of leading NeRF-based parametric face/head models, including MoFaNeRF [247], HeadNeRF [71], and 3DMM-RF [52], as they share our goal of training a general model for face and head reconstruction. We exclude comparisons with StyleGAN-based or diffusion-based methods, as these primarily focus on image synthesis rather than creating a general head model. Since MoFaNeRF and 3DMM-RF only represent the facial region, we compare only the facial region with these methods. For a fair comparison, we use the same set of 550 testing samples as 3DMM-RF, extracted from the CelebAMask-HQ dataset [102]. We assess reconstruction performance using several metrics: pixel-wise L1 distance, LPIPS [234], Structural Similarity Index (SSIM) [202], and Peak Signal-to-Noise Ratio (PSNR). Detailed results of these comparisons are presented in Table 5.2. Additionally, we conduct tests on a separate set of 1,000 images from the FFHQ dataset [87], with outcomes detailed in Table 5.3. Our method demonstrates competitive performance, frequently outperforming these benchmarks in most metrics. Figure 5.7 presents qualitative comparisons, showing that our method achieves a visual quality closely resembling the source image.

We compare our method with HeadNeRF to showcase the superior 3D view

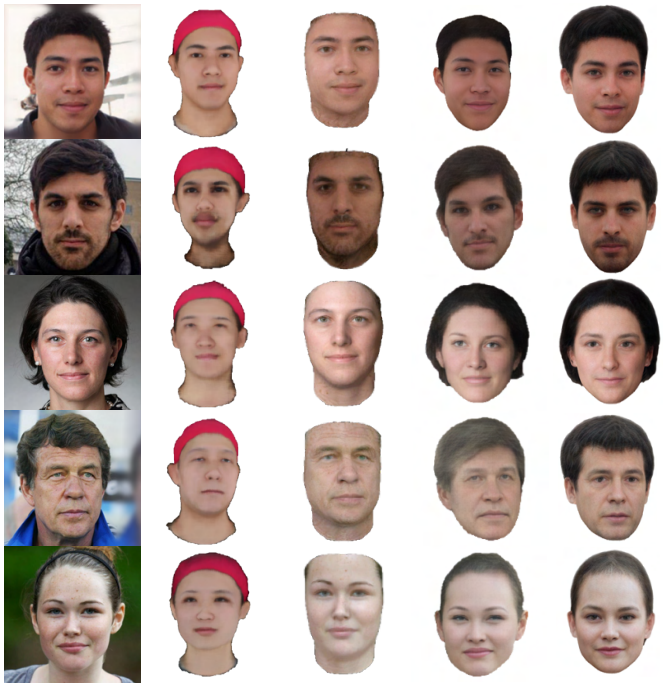


Figure 5.7: Qualitative comparison results on single-image-based reconstruction.

consistency of our 3D Gaussian-based head representation. The results are shown in Figure B.6 in Appendix B.3. As shown, our method consistently outperforms HeadNeRF in maintaining view consistency, which is attributable to the explicit 3D structure represented by the 3D Gaussians. Notably, at extreme viewing angles, where HeadNeRF's renderings exhibit significant deterioration, our 3D Gaussian-based head reconstruction model continues to produce high-quality visual outputs.

Monocular-Video-Based Optimization Results

In the monocular-video-based optimization, we compare our method with the state-of-the-art 3D Gaussian head avatar methods, GaussianAvatars [145] and FlashAvatar [210], both of which are based on the FLAME model. To ensure fairness, we use the same FLAME tracking results to obtain expression coefficients, the un-

Table 5.2: Reconstruction performances (CelebAMask-HQ).

Method	Region	L1 ↓	LPIPS ↓	SSIM ↑	PSNR ↑
MoFaNeRF [247]	Facial	0.273	0.442	0.910	14.713
3DMM-RF* [52]		0.216	-	**0.956**	-
HeadNeRF [71]		0.072	0.113	0.940	24.937
Ours		**0.033**	**0.061**	0.935	**28.755**
HeadNeRF [71]	Head	0.354	0.431	0.693	15.429
Ours		**0.125**	**0.211**	**0.740**	**20.525**

* indicates results reported from the original paper (not open-source).
- indicates the value is missing or not comparable.

Table 5.3: Reconstruction performances (FFHQ).

Method	Region	L1 ↓	LPIPS ↓	SSIM ↑	PSNR ↑
MoFaNeRF [247]	Facial	0.251	0.416	0.720	15.084
HeadNeRF [71]		0.087	0.151	0.920	23.858
Ours		**0.040**	**0.071**	**0.922**	**28.030**
HeadNeRF [71]	Head	0.211	0.310	0.798	18.794
Ours		**0.088**	**0.155**	**0.813**	**23.184**

derlying mesh, and the 6DOF camera pose. Additionally, since FlashAvatar also utilizes a fixed number of 3D Gaussians, we use a similar amount, 74,083, as in our method. GaussianAvatars incorporates dynamic density control from 3DGS, but each learned head model has around 60,000 to 100,000 3D Gaussians, which is similar to ours. Tables 5.4 show the quantitative comparison results. Our method outperforms both GaussianAvatars and FlashAvatar in terms of LPIPS, SSIM, and PSNR metrics across all three subjects, achieving the best results in the shortest training time of only 8 minutes (on a single NVIDIA A6000 GPU). We demonstrate qualitative expression reconstruction results in Figure 5.8, with the first column displaying source expressions that are subsequently applied to the learned head avatars.

We compare convergence speed with FlashAvatar, known for its fast training. Figure 5.9 displays the comparison. Our approach, utilizing a better initialization and a non-neural network strategy, achieves faster convergence than FlashAvatar. Ultimately, our method enables the efficient optimization of a high-density

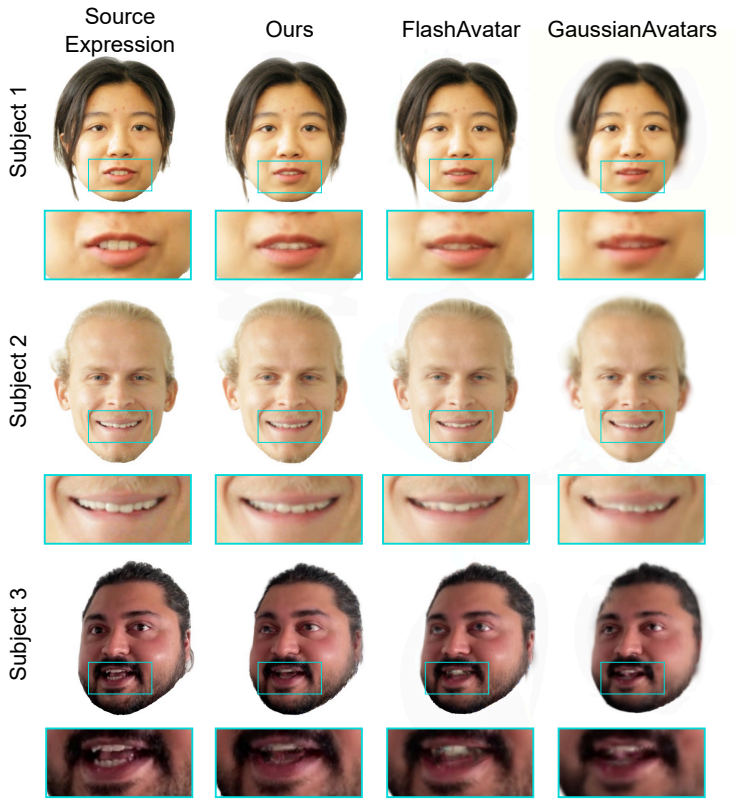


Figure 5.8: Self-reenactment results.

3D Gaussian head avatar within minutes.

Ablation Study 1: Training Dataset for Single-Image Reconstruction Model

We conduct an ablation study on the training dataset used for training our single-image-based reconstruction model. Our complete method uses both synthetic and real data for training. In this ablation study experiment, we train a reconstruction model solely on the synthetic dataset. Table 5.5 and Table 5.6 show the evaluation results on the CelebAMask-HQ dataset and the FFHQ dataset, respectively. We observe a noticeable degradation in the reconstruction quality of real images when training is performed without using real data. This outcome highlights a discrepancy between synthetic and real data, underscoring the essential role of real-image

Table 5.4: Average quantitative comparison across all three subjects of the PointAvatar dataset.

Method	Train Time	LPIPS ↓	SSIM ↑	PSNR ↑
GaussianAvatars (CVPR'24) [145]	3 hrs	0.1437	0.9010	24.4913
GaussianAvatars (CVPR'24) [145]	6 hrs	0.1433	0.9023	24.7313
FlashAvatar (CVPR'24) [210]	30 mins	0.0970	0.8853	24.6800
FlashAvatar (CVPR'24) [210]	60 mins	0.0930	0.8887	24.7670
Ours (global rectification)	8 mins	0.0773	0.8990	24.5757
Ours	8 mins	**0.0700**	**0.9113**	**25.3580**



Figure 5.9: Qualitative comparison of the convergence speed of video-based optimization.

training in achieving accurate reconstructions.

Ablation Study 2: Regularization Loss Terms

We first use an example to demonstrate the importance of our view consistency regularization loss $\mathcal{L}_{\text{view}}$, see Figure 5.10. The first row displays input images of the same identity from different camera views fed into our single-image-based reconstruction model. The second (w/o $\mathcal{L}_{\text{view}}$) and third (w/ $\mathcal{L}_{\text{view}}$) rows show the

Table 5.5: Ablation Study on Training Data: Reconstruction performances (CelebAMask-HQ).

Method	Region	L1 ↓	LPIPS ↓	SSIM ↑	PSNR ↑
Ours (Synthetic Data)	Facial	0.043	0.072	0.922	26.411
Ours (Synthetic + Real Data)		**0.033**	**0.061**	**0.935**	**28.755**
Ours (Synthetic Data)	Head	0.188	0.250	0.692	16.769
Ours (Synthetic + Real Data)		**0.125**	**0.211**	**0.740**	**20.525**

Table 5.6: Ablation Study on Training Data: Reconstruction performances (FFHQ).

Method	Region	L1 ↓	LPIPS ↓	SSIM ↑	PSNR ↑
Ours (Synthetic Data)	Facial	0.061	0.092	0.896	23.707
Ours (Synthetic + Real Data)		**0.040**	**0.071**	**0.922**	**28.030**
Ours (Synthetic Data)	Head	0.178	0.213	0.747	17.157
Ours (Synthetic + Real Data)		**0.088**	**0.155**	**0.813**	**23.184**

reconstructed heads rendered from a frontal view. As evident, without $\mathcal{L}_{\text{view}}$, the reconstruction quality is biased towards the side visible in the input image. However, with $\mathcal{L}_{\text{view}}$, the reconstructions appear nearly identical across different views of the same identity, showcasing the robustness of our approach in maintaining consistent visual outputs.

We demonstrate the critical role of our 3D Gaussian position and scale regularization losses in Figure 5.11. In this experiment, we trained two additional single-image-based reconstruction models, each omitting either the $\mathcal{L}_{\mu}$ (position loss) or $\mathcal{L}_{\mathbf{s}}$ (scale loss). These models were used to initialize 3D Gaussian heads, which were then further optimized using video data. The results clearly show that without adequate regularization of the position and scale of the 3D Gaussians, noticeable artifacts persist even after subsequent video-based optimization.

In Figure 5.12, we showcase the importance of our expression-aware blendmaps. We apply the expression estimated from the source image to our learned head avatars. The second image displays the head enhanced with our expression-aware UV blendmaps, and the third illustrates the face with only global UV rectification. Both approaches use the same amount of time for optimization in this ablation experiment. It becomes clear that global UV rectification introduces artifacts when



Figure 5.10: Ablation results on view consistency regularization.

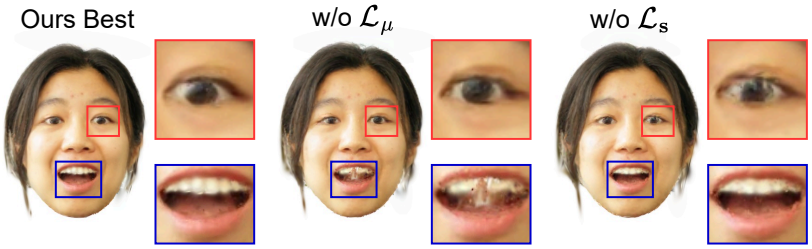


Figure 5.11: Ablation results on position and scale regularizations.

dealing with exaggerated expressions. Quantitative results, which further substantiate the limitations of using only global rectification, are presented in Table 5.4.

Ablation Study 3: UV Position Regularization Weights

We use an ablation experiment to demonstrate the importance of our UV position regularization weights mask (shown in Figure 5.3). In this experiment, we retrain a reconstruction model with uniform position regularization weights across all 3D Gaussians. Figure 5.13 shows the comparisons between our original model (Best Result) and the retrained model (Uniform Weight). As shown, if we use uniform

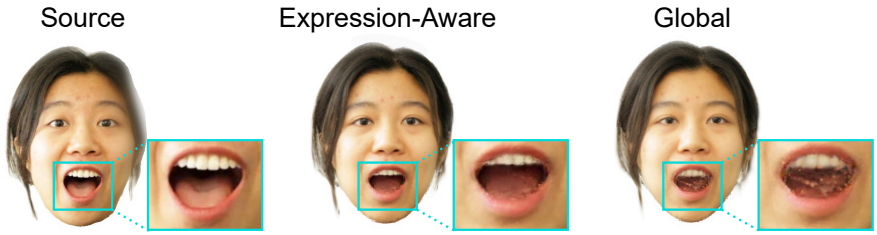


Figure 5.12: Ablation results on different levels of rectification.

position regularization weights, the 3D Gaussians on the scalp are constrained from moving too far, and thus the scales of these 3D Gaussians are enlarged to match the hair. However, this will result in the loss of fine details and the appearance of blurry artifacts. If we manually reduce the scales of each 3D Gaussian, we can observe that by giving more Weight to the facial region while less Weight to the rest (Best Result), we give non-facial-region 3D Gaussians more freedom to move and form the shape of hair.

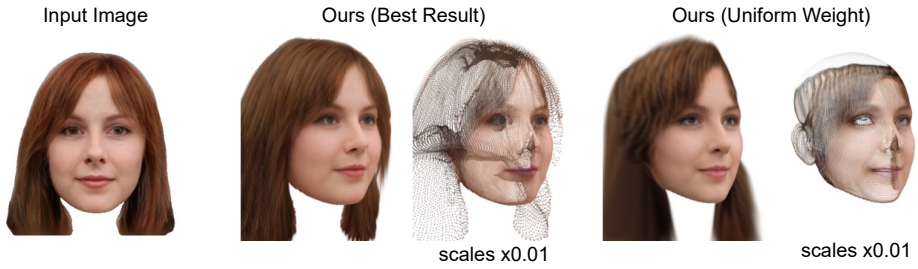


Figure 5.13: Ablation study on UV position regularization weights. The right images are the renderings of the reconstructed 3D Gaussian head of the input image.

Ablation Study 4: Initialization Schemes in Personalization

We test initializing the 3D Gaussians with FLAME reconstruction's shape (FLAME-initialized). The convergence during training is shown in Figure 5.14. As illustrated, simply initializing the locations of 3D Gaussians without properly initializing other Gaussian parameters leads to slow convergence.



Figure 5.14: Convergence with different initialization schemes.

5.3.4 Limitations

We found that the highest quality achievable by Déjà-vu relies significantly on the accuracy of the FLAME reconstruction. Additionally, our current approach does not account for lighting variations, so if the personal video was recorded under changing lighting conditions, the resulting head avatar may exhibit visual artifacts.

5.3.5 Summary

In this section, we propose Déjà-vu, a novel framework for creating controllable 3D Gaussian head avatars with fast training. Notably, Déjà-vu is the first method to reconstruct a 3D Gaussian head solely from a single input image. This reconstruction model provides robust initialization for our video-based optimization. We propose learnable UV blendmaps to adjust the head to have the desired expression, a solution that is both effective and quick to train. Our framework outperforms existing state-of-the-art methods in both rendering quality and training speed. In the future, we will enhance Déjà-vu's adaptability to a broader range of facial expressions and explore additional applications.

5.4 Method 2: Continuous Level of Detail Control for 3D Gaussian Head Avatars (ArchitectHead)

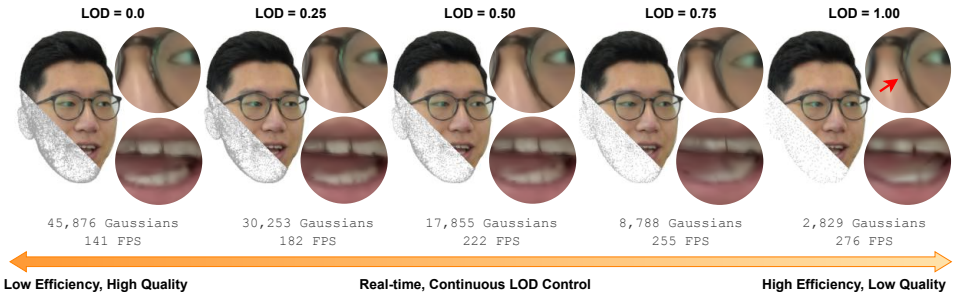


Figure 5.15: ArchitectHead supports continuous LOD control ranging from 0 (highest) to 1.0 (lowest). This figure shows the rendered image quality under different LOD settings. For each case, we provide a zoom-in view of selected regions. The grey dots indicate the positions of the 3D Gaussian points. The test identity is taken from the NeRFace dataset [51]. The red arrow indicates visible artifacts (sparse Gaussians) in the lowest LOD.

5.4.1 Overview

3D Gaussian Splatting (3DGS) has enabled photorealistic and real-time rendering of 3D head avatars. Existing 3DGS-based avatars typically rely on tens of thousands of 3D Gaussian points (Gaussians), with the number of Gaussians fixed after training. However, many practical applications require adjustable levels of detail (LOD) to balance rendering efficiency and visual quality. Existing methods typically bind the 3D Gaussians to a mesh-based head model to provide controllability, and a natural way to control the number of Gaussians is to use a 2D UV map as an intermediate representation, where the UV map resolution determines the Gaussian count. However, two challenges remain in this UV-based strategy. First, the UV position map alone does not capture sufficient local information to represent detailed 3D head appearance. Second, it is necessary to balance different resolutions while maintaining smooth transitions across LODs.

In this section, we propose "ArchitectHead", the first framework for creating

3D Gaussian head avatars that support continuous control over LOD. Our key idea is to parameterize the Gaussians in a 2D UV feature space and propose a UV feature field composed of multi-level learnable feature maps to encode their latent features. A lightweight neural network-based decoder then transforms these latent features into 3D Gaussian attributes for rendering. ArchitectHead controls the number of Gaussians by dynamically resampling feature maps from the UV feature field at the desired resolutions. We formulate continuous LOD control as resampling from the 2D UV feature space, making controlling the number of Gaussians in a head avatar as simple as resizing a 2D image. To achieve this, we design a learnable multi-level UV feature field, structured as a pyramid of UV-anchored feature maps at different resolutions. The feature maps store the per-Gaussian latent features learned during training. Our model is trained per-person, following the training objective of existing monocular head-avatar works to achieve better personalization [29, 41, 108, 210]. After training, we can sample a UV feature map of any resolution between the maximum and minimum levels to achieve runtime LOD control. Our sampling strategy assigns higher weights to feature maps with resolutions closer to the desired one to blend the UV feature maps of different resolutions, ensuring smooth transitions across LODs. The sampled features are concatenated with positionally encoded point locations, the expression code, and the LOD value to form the latent representation of the Gaussian points. We employ a lightweight neural decoder to convert these latent features into 3D Gaussian attributes, ready for rendering.

The training has two stages. In the first stage, we train with the highest LOD (LOD 0) to capture fine details and stabilize learning. In the second stage, we randomly sample LOD values between 0.0 and 1.0 so the decoder can adapt to different levels of detail. This design enables efficient and flexible LOD control for 3D Gaussian head avatars without the need for retraining. Figure 5.15 shows an example of our reconstructed head avatar with different LOD settings. Experimental results show that ArchitectHead achieves state-of-the-art (SOTA) quality in self- and cross-identity reenactment tasks at the highest LOD, while maintaining near-SOTA performance at lower LODs. At the lowest LOD, our method uses only 6.2% of the Gaussians while the quality degrades moderately (L1 Loss +7.9%, PSNR –0.97%, SSIM –0.6%, LPIPS Loss +24.1%), and the rendering speed nearly

doubles.

The main contributions of this work are:

1. We propose ArchitectHead, a novel framework for creating 3D Gaussian head avatars that support on-the-fly LOD control without retraining. This is the first 3D Gaussian head avatar method to enable real-time, continuous LOD adjustment, providing greater scalability for balancing rendering quality and efficiency across applications.

2. We introduce a learnable multi-level UV feature field, along with an adaptive sampling strategy, to capture spatial information across high-to-low resolutions. This ensures smooth transitions between different LODs.

3. We conduct extensive experiments on monocular video datasets for 3D head avatar reconstruction, demonstrating that ArchitectHead achieves effective, scalable control over LOD while maintaining photorealistic quality.

5.4.2 Background

Comparison with FlashAvatar

UV-based FlashAvatar [210] is similar to our method. It uses a multi-layer perceptron (MLP) decoder to generate expression-conditioned 3D Gaussians by taking a positionally encoded UV mesh geometry map and expression coefficients from FLAME [109] as input. Despite its simple yet effective solution, it does not support changing the UV map resolution after training. This is because 3D Gaussians hold complex information, including unique rotations and sizes, which prevents them from being easily scaled like images. Our work addresses this by proposing a learnable and scalable UV-based feature field that enables the decoder to adapt to varying UV resolutions. While the broader landscape of 3D Gaussian-based head avatars is reviewed in Section 5.2 (in the context of Déjà-vu), we note that a common limitation persists across all these methods: they use a fixed number of Gaussians after training, making Level of Detail (LOD) control challenging. Our work is the first to introduce continuous LOD control for 3DGS head avatars, enabling a dynamic balance between rendering efficiency and visual quality to suit

diverse practical needs.

Level of Detail in 3D Gaussian Splatting

Recent research has extended LOD concepts to 3DGS, but these methods are primarily for static scenes, such as urban environments [117, 205]. Some methods introduce hierarchical structures, enabling efficient rendering by pruning distant or less important Gaussians [96, 149, 160]. Other methods, such as CityGaussian and FLoD, focus on scalable multi-level representations to adapt to hardware constraints [117, 156]. LOD-GS applies a filtering mechanism sensitive to the sampling rate to improve visual quality across different zoom levels [223]. Milef et al. propose a continuous LOD method that learns to rank splats by their importance, enabling efficient distance-based rendering without inference-time overhead [129]. Although these methods are effective for large static scenes, they are less suitable for 3D head avatars, which are highly dynamic and require controllability to support facial animation.

The most relevant method to ours is LoDAvatar, which adapts mesh triangle-bound Gaussians for full-body avatars using hierarchical embedding and selective detail enhancement [42]. However, LoDAvatar only supports discrete LOD control and relies on synthetic multi-view images for training, making it less suitable for personalized avatar creation from monocular video. In contrast, our work supports continuous LOD control for 3D Gaussian head avatars, making LOD changes smooth while enabling training from monocular videos for better photorealism.

5.4.3 The Proposed Method

Framework Overview

As shown in Figure 5.16, we leverage the geometry prior of the FLAME head model [109] to initialize the positions of 3D Gaussians. The 3D head mesh m is derived as $m = \mathcal{F}(\beta_{id}, \beta_{exp}, \beta_{jaw}, \beta_{eye})$, where $\mathcal{F}$ is the FLAME decoder, $\beta_{id} \in \mathbb{R}^{300}$, $\beta_{exp} \in \mathbb{R}^{100}$, $\beta_{jaw} \in \mathbb{R}^{3}$, and $\beta_{eye} \in \mathbb{R}^{6}$ are the identity code, expression code, jaw pose, and eye poses, respectively. Following [145], we add teeth to the original FLAME mesh to better initialize Gaussians in the mouth region. The mesh

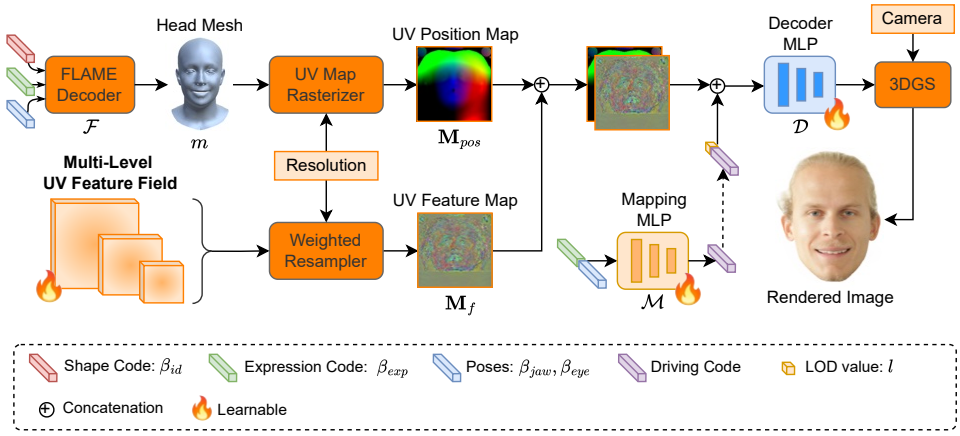


Figure 5.16: Pipeline of ArchitectHead. We propose a 3D Gaussian head avatar creation method with continuous level of detail (LOD) control. Starting from shape and expression codes, we use the FLAME head model to generate the 3D mesh geometry, which is rasterized into a UV position map at the desired resolution. A multi-level UV feature field is introduced to learn local latent features, from which our weighted resampler extracts a UV feature map of the target resolution. This map is concatenated with the UV position map, the desired LOD value, and a driving code obtained from expression and pose codes via an MLP network $\mathcal{M}$. The resulting pixel-wise latent features are decoded by an MLP-based decoder into 3D Gaussian attributes, which are rendered using 3D Gaussian Splatting (3DGS).

geometry is rasterized into a UV map. We then apply positional encoding following [210] to obtain $\mathbf{M}_{pos} \in \mathbb{R}^{S \times S \times D_{PE}}$, where S is the desired UV resolution and D_{PE} is the dimensionality after positional encoding.

Denote the continuous LOD level as $0 \leq l \leq 1.0$, where $l = 0$ corresponds to the highest LOD and $l = 1.0$ to the lowest. Also, define S_{max} and S_{min} as the maximum and minimum UV map resolutions, respectively. The desired UV map resolution is then computed as:

$$S = S_{\max} - l\,(S_{\max} - S_{\min}). \tag{5.9}$$

The UV feature map $\mathbf{M}_f \in \mathbb{R}^{S \times S \times D_f}$, where D_f is the feature dimension, is resampled from the proposed multi-level UV feature field (detailed in

Section 5.4.3). This map is then concatenated channel-wise with a positionally encoded UV position map, $\mathbf{M}_{pos}$, to inject various frequency-based information into the features. This approach, leveraging the learnable feature map $\mathbf{M}_f$, significantly improves the model's ability to represent local details, thereby enhancing visual quality.

Inspired by [108], we use a small multi-layer perceptron (MLP) network $\mathcal{M}$ to learn the mapping from higher dimension codes (β_{exp}, β_{jaw}, and β_{eye}) to a lower dimension driving code with K dimensions. This code conditions the decoder network on facial expressions. The mapping helps the model focus on the most important expression variations and reduces overfitting to specific expression or pose codes. We then concatenate the driving code with the LOD value l to form a one-dimensional vector. This vector is then appended to each pixel of the UV map, resulting in a final feature map with $D_{PE} + D_f + K + 1$ channels.

The decoder network consists of five MLPs, each used to decode the per-Gaussian UV feature into one of the 3D Gaussian attributes: $\Delta\mu, \mathbf{s}, \mathbf{q}, \alpha, \mathbf{c}$. The final Gaussian location is computed as $\mu = \Delta\mu + \mu_{init}$, where μ_{init} is the initial position from the UV position map. Given the camera pose, the decoded 3D Gaussians are rendered through the 3DGS rasterizer to derive the image. Please refer to the supplementary material for details.

Multi-Level UV Feature Field

Existing works have shown that using higher-dimensional learnable 3D Gaussian features can produce better visual quality [61, 74, 158]. This is because higher-dimensional features provide each 3D Gaussian point with greater representational capacity. Motivated by this, we propose using a 2D UV feature map to encode and learn these high-dimensional, per-Gaussian features. However, this approach presents a challenge when implementing continuously controllable LOD if the UV feature map is simply resized. Since each pixel in the map corresponds to a 3D Gaussian point, neighboring Gaussians can have significantly different rotations and scales. While downsampling from a high to a low resolution can merge features of nearby Gaussians, it also smooths out critical information, causing adjacent Gaussians to share overly similar attributes and compromising local detail.

To address the above-mentioned issue, we propose to use multiple UV feature maps of different resolutions to form a multi-level feature field. We can resample a UV feature map of any size within the range of the largest and smallest UV feature maps in the multi-level feature field. Define the feature field as a set of UV feature maps $\mathcal{U} = \{\mathbf{M}_f^{S_1}, ..., \mathbf{M}_f^{S_N}\}$ ordered by the resolutions $S_1, ..., S_N$ increasingly. $\mathcal{U}$ contains at least the feature maps of the maximum and minimum resolutions. To sample the feature map of a given resolution S ($S_{min} \leq S \leq S_{max}$), we first resize all the feature maps to the resolution of S using bi-cubic interpolation. Then we blend all the resized feature maps via the following formula:

$$\mathbf{M}_f = \sum_{i=1}^{N} w_i * \mathcal{I}(\mathbf{M}_f^{S_i}, S), \tag{5.10}$$

where w_i is the blending weight for the i^{th} feature map $\mathbf{M}_f^{S_i}$, and $\mathcal{I}(\mathbf{M}_f^{S_i}, S)$ is the bi-cubic interpolation function that resizes $\mathbf{M}_f^{S_i}$ to a given resolution S.

The computation of w_i is defined as follows:

$$w_i = \frac{\exp(-|r_i - r|/\tau)}{\sum_{j=1}^{N} \exp(-|r_j - r|/\tau)}, \tag{5.11}$$

where $r_i = log_e S_i$, $r = log_e S$, $\tau > 0$ is the temperature controlling the softness of interpolation. Equation 5.11 is a softmax-based formulation, which gives higher weight to the feature map whose resolution is closer to S, while ensuring w_i never reaches zero so that gradients can always propagate properly.

Loss Functions

The training loss is defined as:

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_{lpips}\mathcal{L}_{lpips} + \lambda_{\mu}\mathcal{L}_{\mu} + \lambda_{\mathbf{s}}(1 - 0.5l)\mathcal{L}_{\mathbf{s}}, \tag{5.12}$$

where $\mathcal{L}_{rgb}$ is the image reconstruction loss, $\mathcal{L}_{lpips}$ is the LPIPS perceptual loss [210, 234], $\mathcal{L}_{\mu}$ and $\mathcal{L}_{\mathbf{s}}$ are regularization terms on the Gaussian point location offsets and scales, respectively, and λs are their weights. Both $\mathcal{L}_{\mu}$ and $\mathcal{L}_{\mathbf{s}}$ are computed using the mean L2 norm of the location offsets and scales. For $\mathcal{L}_{\mathbf{s}}$, we

assign more weights to smaller l (higher LOD) because the Gaussians are denser. Specifically, $\mathcal{L}_{rgb}$ has two sub-terms:

$$\mathcal{L}_{rgb} = \mathcal{L}_H(I, \hat{I}) + \lambda_{parts}\mathcal{L}_H(I \cdot \mathbf{B}, \hat{I} \cdot \mathbf{B}), \tag{5.13}$$

where $\mathcal{L}_H$ is the Huber loss [77], I is the ground-truth image, $\hat{I}$ is the rendered image, and $\mathbf{B}$ is the binary mask of eyes and mouth parts.

Training Scheme

The training has two stages. In the first stage, we use the highest Level of Detail (LOD) to jointly optimize the multi-level latent field along with the mapping network, $\mathcal{M}$, and the decoder network $\mathcal{D}$. The second stage of training focuses on the model's ability to handle continuous detail. In this stage, we randomly sample LODs from the range $[0, 1.0]$ and accumulate their loss gradients for optimization, while keeping the mapping network $\mathcal{M}$ fixed.

5.4.4 Experiments

Datasets

We use two monocular video datasets in our experiments: the PointAvatar dataset [239] and the INSTA dataset [249]. The PointAvatar dataset contains videos of three subjects, while the INSTA dataset provides ten videos of different subjects. Since one subject in INSTA overlaps with PointAvatar, we use only nine videos from INSTA. For each video, the first 90% of frames are used for training and the remaining 10% for evaluation.

Implementation Details

We implement our method using *PyTorch*. To render the 3D Gaussians, we use the open-source 3DGS rasterizer *gsplat* [224]. The maximum and minimum UV map resolutions are set to $S_{max} = 256$ and $S_{max} = 64$, respectively. We use 12 frequencies for both sine and cosine in the positional encoding, resulting in $D_{pos} = 75$ dimensions (see supplementary materials). Following [108], we set

the reduced driving code dimension to $K = 20$. The learnable feature maps have $D_f = 64$ channels. The multi-level feature field consists of three resolutions, 256×256, 128×128, and 64×64, respectively. The temperature in Equation 5.11 is set to $\tau = 0.35$. We use $\mathbf{SH} = 3$ in our work. We implement a fitting-based FLAME head tracker to derive the FLAME codes and camera pose for each frame in the monocular videos.

Training: Training is conducted on a single NVIDIA RTX 4090 GPU with a batch size of one. In stage one, we train for 15,000 steps. In stage two, we render five LOD levels at each step and train for 30,000 steps in total. The training time is approximately 30 minutes for stage one and 2.5 hours for stage two. Please refer to supplementary materials for details. The loss term weights are $\lambda_{parts} = 20$, $\lambda_{lpips} = 0.05$, $\lambda_{\mu} = 0.001$, and $\lambda_{\mathbf{s}} = 0.5$.

Baselines

The baselines are state-of-the-art Gaussian head avatar methods [108, 145, 210, 219]. Both GaussianAvatars [145] and RGBAvatar [108] bind Gaussians to triangle facets, while FlashAvatar [210] and Gaussian Déjà-vu [219] adopt UV-based representations. In addition, both RGBAvatar and Gaussian Déjà-vu employ Gaussian blendshapes to achieve expressive animation. These methods are representative of the major categories of Gaussian head avatars, including mesh-bound [108, 145], UV-based [210, 219], and blendshape-based [108, 219].

Self- and Cross-Identity Reenactment

Self-reenactment. The self-reenactment task involves using FLAME codes from a specific identity to drive an avatar model that was trained on that same identity's video. We conduct experiments on both the PointAvatar and INSTA datasets. The quantitative results in Tables 5.7 and 5.8 show that the proposed method achieves the best performance on both datasets across all metrics, including L1 distance, PSNR, SSIM, and perceptual loss (LPIPS). We also evaluated the trained avatar in half-precision (fp16), and the results are comparable to those obtained with full precision. Figure 5.17 presents qualitative results. At the highest LOD setting, our method captures finer details, while at the lowest LOD setting it can still repre-



Figure 5.17: Qualitative comparisons of self-reenactment results. Selected regions are zoomed in for clearer comparison of fine details. The last two columns show our method with the highest (LOD=0.0) and lowest (LOD=1.0) settings. Compared to existing methods, our approach preserves finer details at the highest LOD while also maintaining reasonable quality at the lowest LOD.

sent reasonable head shapes with sparse Gaussian points. In addition, Figure 5.18 illustrates novel views rendered from the reconstructed avatars.

Table 5.7: Quantitative comparisons of self-reenactment results on PointAvatar dataset.

Method	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓
GaussianAvatars [145]	0.048	26.009	0.916	0.112
FlashAvatar [210]	0.044	27.223	0.915	0.070
RGBAvatar [108]	0.053	28.106	0.907	0.100
Gaussian Déjà-vu [219]	0.045	27.819	0.923	0.065
Ours (fp16)	**0.038**	28.615	**0.928**	**0.058**
Ours (best)	**0.038**	**28.621**	**0.928**	**0.058**

Table 5.8: Quantitative comparisons of self-reenactment results on INSTA dataset.

Method	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓
GaussianAvatars [145]	0.030	26.979	0.945	0.069
FlashAvatar [210]	0.027	28.362	0.947	0.039
RGBAvatar [108]	0.037	27.583	0.924	0.060
Gaussian Déjà-vu [219]	0.024	29.722	0.956	0.034
Ours (fp16)	**0.022**	30.379	**0.960**	**0.032**
Ours (best)	**0.022**	**30.389**	**0.960**	**0.032**

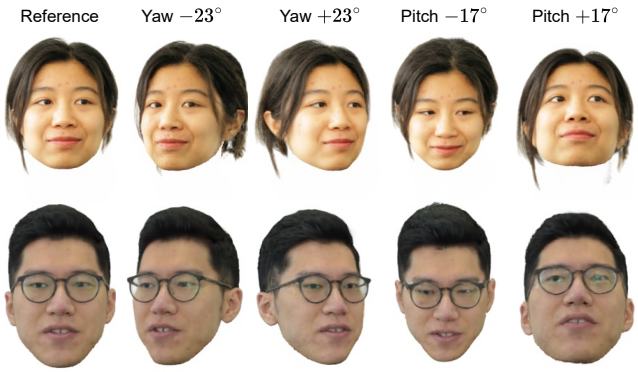


Figure 5.18: Rendered novel views. The reference images (left-most) are rendered with the default camera pose, while the other images are rendered with yaw or pitch angle offsets using an orbit camera.

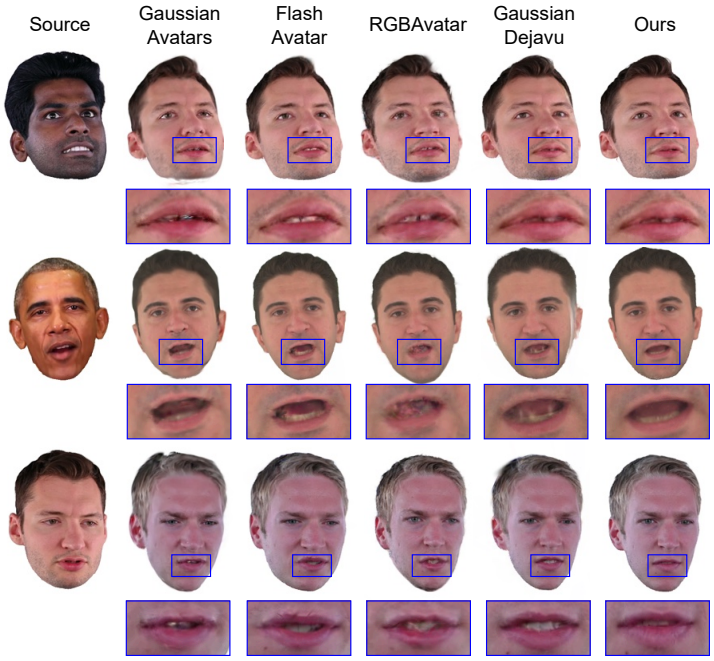


Figure 5.19: Qualitative results of cross-identity reenactment. The first column shows the source images that provide the expression codes and camera poses. The remaining columns in each row present the trained head avatar of another individual reenacted using the source codes and poses.

Cross-Identity Reenactment. In cross-identity reenactment, we use FLAME codes from the video of one identity to drive avatars trained on another identity. Figure 5.19 shows qualitative results. Our method achieves higher fidelity and better visual quality compared to the baselines.

Level of Detail Control

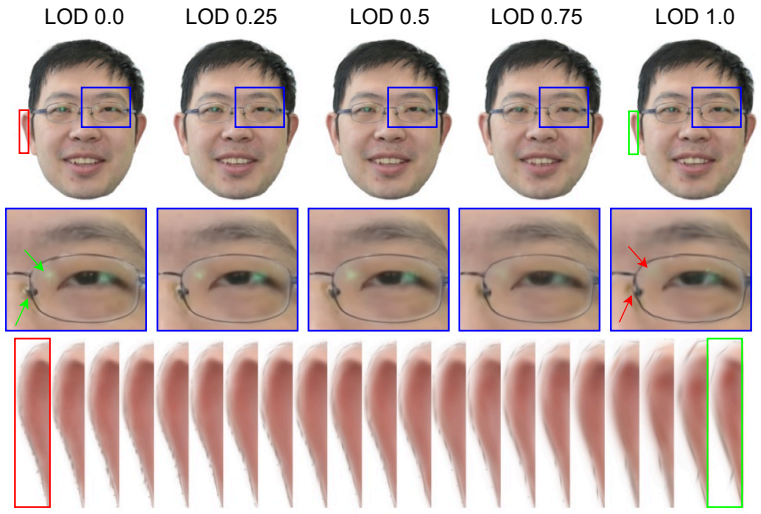


Figure 5.20: Qualitative results of varying LODs. The second row shows zoomed-in regions highlighted by the blue rectangle in the first row. Red arrows indicate sub-optimal results (missing glare and fine details), compared to the corresponding locations marked by green arrows in our highest LOD (LOD=0) result. The third row illustrates the same region (ear) rendered with different LODs from 0.0 to 1.0 in steps of 0.05.

In this experiment, we demonstrate the proposed method's LOD control capability. Figure 5.20 shows the trained avatar model rendered under different LOD settings. We observe that while the overall differences are subtle, the quality decreases in zoomed-in regions at lower LODs. Therefore, for scenes where the camera is close to the avatar, we recommend using a higher LOD to achieve better visual quality. Conversely, if the camera is far away or the quality is less critical, a lower LOD can be used to save computational resources. We also benchmark

the rendering speed across different LOD levels, with results summarized in Table 5.9. Using half-precision (fp16) leads to significantly faster rendering at higher LODs. Importantly, the rendering quality does not degrade noticeably with fp16, even though the model is trained in full precision (fp32); detailed comparisons are provided in the supplementary materials.

Table 5.9: Rendering FPS ↑ (frames per second) performance comparison across LOD settings, GPUs, and floating-point precisions. *#Gaussians* denotes the number of 3D Gaussian points.

LOD	#Gaussians	4090 (fp16)	4090 (fp32)	A6000 (fp16)	A6000 (fp32)
0.00	45,876	141	57	64	38
0.25	30,253	182	132	86	61
0.50	17,855	222	181	113	87
0.75	8,788	255	245	146	123
1.00	2,829	276	301	142	160

In Table 5.10, we evaluate different LOD settings. At the lowest LOD ($l = 1.0$), the model uses only 6.2% of the Gaussians compared to the highest LOD, with a slight increase in L1 loss (+7.9%), and minor drops in PSNR (–0.97%) and SSIM (–0.6%), while nearly doubling rendering speed (+96% FPS).

Ablation Studies

We conduct ablation studies on the UV feature field. In the first ablation, we completely remove the UV feature field, meaning no learnable per-Gaussian features (*w/o fmap*). In the second ablation, we replace the multi-level feature field with a single learnable feature map, and test three resolutions: 64×64, 128×128, and 256×256. Self-reenactment experiments are performed on the PointAvatar dataset, with results summarized in Table 5.10. We observe that without learnable features, the performance is the worst. When using a single-resolution feature map, higher resolution produces better quality. However, the proposed multi-level feature field (*full*) achieves the best results in most cases.

In Figure 5.21, we present the qualitative ablation results. Notably, when comparing *fmap (256)* with our best result, we observe more visual artifacts and blurrier sharp edges in *fmap (256)*. We attribute this to the fact that a single feature map

Table 5.10: Quantitative comparison across different ablation settings and LODs. The **bold** results indicate the globally best score across all methods and LODs. Within each LOD setting (0.0, 0.5, 1.0): we color the best (orange) and the second best (lighter orange) results. *w/o fmap* denotes our method without the learnable UV feature map. *fmap (.)* denotes our method where the multi-level UV feature field is replaced by a fixed-resolution learnable feature map, with the value in parentheses indicating the resolution of the feature map.

Method	LOD	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Ours (full)	0.0	**0.038**	**28.621**	**0.928**	**0.058**
	0.5	0.039	28.514	0.927	0.061
	1.0	0.041	28.342	0.922	0.072
w/o fmap	0.0	0.044	27.815	0.913	0.074
	0.5	0.044	27.864	0.914	0.074
	1.0	0.048	27.436	0.907	0.087
fmap (64)	0.0	0.041	28.295	0.913	0.063
	0.5	0.040	28.391	0.915	0.065
	1.0	0.041	28.321	0.921	0.075
fmap (128)	0.0	0.041	28.289	0.916	0.062
	0.5	0.041	28.423	0.917	0.065
	1.0	0.042	28.191	0.912	0.074
fmap (256)	0.0	0.039	28.614	0.926	0.059
	0.5	0.039	28.411	0.924	0.064
	1.0	0.041	28.323	0.923	0.074

must represent all LOD levels, causing the lower LOD representations to negatively affect the higher LOD results.

5.4.5 Limitations

Like most existing methods, our method relies on accurate FLAME tracking to achieve reliable 3D-2D alignment, which is essential for maintaining 3D consistency. In addition, we observe that some expression modes appear only under large head poses in the video (e.g., side views). When this happens, the network tends to overfit these rare cases, leading to artifacts when such expression modes occur.

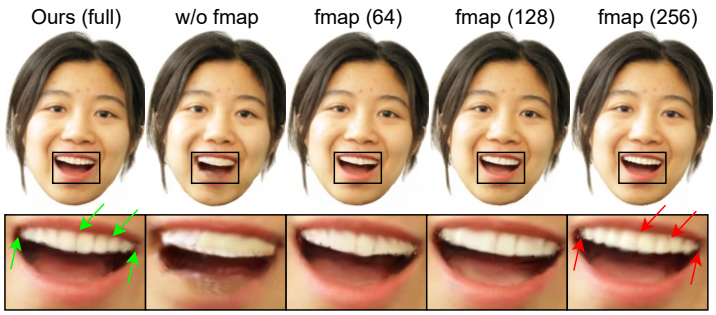


Figure 5.21: Qualitative results of ablation study. *w/o fmap* denotes our method without the learnable UV feature map. *fmap (.)* denotes our method where the multi-level UV feature field is replaced by a fixed-resolution learnable feature map, with the value in parentheses indicating the resolution of the feature map. Red arrows indicate artifacts, compared to the corresponding locations marked by green arrows in our best (full method) result. We recommend zooming in to better inspect the details.

5.4.6 Summary

In this section, we propose ArchitectHead, a framework for creating 3D Gaussian head avatars that supports real-time and continuous adjustment of the level of detail (LOD). To our knowledge, ArchitectHead is the first 3D Gaussian head method to realize continuous LOD control. We parameterize Gaussians in the UV feature space, allowing us to control their number by simply adjusting the UV map resolution. A neural decoder then generates the Gaussian attributes, using the LOD as an additional condition. To capture rich local information and balance different LODs while ensuring smooth transitions between them, we introduce a learnable UV latent feature map alongside the UV position map, providing more representative information. We then extend this design to a multi-level latent feature field, which enables weighted resampling across resolutions and improves the balance among varying LODs. Experiments on monocular video datasets show that ArchitectHead achieves state-of-the-art quality. For future work, we plan to extend the framework to multi-view video datasets and improve training efficiency.

5.5 Method 3: FOV Estimation for 3D Face Reconstruction — An Application of 3DGS-Based Head Models in Synthetic Data Generation

5.5.1 Overview

Existing image-based 3D face reconstruction methods rely on a virtual camera to project the reconstructed 3D face onto the 2D image plane for comparison with the input image, a crucial step for accurate results. To simplify the reconstruction process, these methods often use fixed camera intrinsics and assume minimal perspective distortion, overlooking the varying distortion levels in in-the-wild images and leading to inaccuracies in reconstructed 3D face shapes. To address this issue, we propose estimating the virtual camera's optimal field-of-view (FOV) for a given image, enabling consistent 3D face reconstruction across varying distortion levels. We introduce two synthetic datasets: one for training our FOV estimation network (FOV-Net) and another for evaluating its performance and reconstruction accuracy. We use the estimated FOV to initialize the camera for the fitting-based reconstruction process. Experiments show that our approach significantly improves reconstruction consistency under different levels of perspective distortion.

In summary, our work makes the following main contributions:

1. We propose estimating the virtual camera's FOV to address shape inconsistencies caused by perspective distortion in image-based 3D face reconstruction;

2. We introduce FOV-Net, a neural network model designed to estimate camera FOV from face images;

3. We present two new datasets, PanoHead-FOV and NeRSemble-FOV, designed explicitly for training FOV-Net and evaluating 3D face reconstruction accuracy under varying levels of perspective distortion.

5.5.2 Background

To account for perspective effects, several works have estimated camera intrinsic parameters as part of 3D face reconstruction [73, 126]. Merras et al. compute the camera projection matrix using facial landmarks and multiple images taken from different angles, but provide only rough estimates due to ambiguities [126]. Hu et al. propose a self-calibration method that alternates between optimizing camera parameters and refining 3D face shapes; however, it requires video frames and assumes a consistent focal length across all frames [73]. Other methods assume fixed camera intrinsics and instead attribute perspective distortion to camera-to-face distance [85, 230]. Zhang et al.'s method requires the original image size to remain unchanged, failing when images are cropped [230], while 6DoF-Face relies on a costly internal dataset acquired using iPhones for training [85]. In contrast, our method directly estimates the camera FOV from a single image and does not require multi-view input, video sequences, fixed image sizes, or proprietary training data.

5.5.3 The Proposed Method

Perspective Camera Model

We parameterize our perspective camera model using intrinsic and extrinsic matrices. This virtual camera model is used in rendering the reconstructed 3D head into a 2D image. The projection matrix $M \in \mathbb{R}^{3\times4}$ is defined as:

$$M = \underbrace{\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic}} \cdot \underbrace{\Big[R|T\Big]}_{\text{Extrinsic}}, \tag{5.14}$$

where f is the focal length, (c_x, c_y) is the principal point, $R \in SO(3)$ is the rotation matrix representing camera orientation with 3 degrees of freedom (DoF), and $T \in \mathbb{R}^{3\times1}$ is the translation vector representing camera position with 3DoF. Note that, both f and (c_x, c_y) are measured in pixels.

We assume the image has equal width and height of s pixels and $c_x = c_y =$

$s/2$. Accordingly, the relationship between the focal length and the camera's effective FOV θ, measured in degrees, is given by:

$$\theta = 2 \cdot \textit{arctan}(\frac{s}{2f}) \cdot \frac{180}{\pi}. \tag{5.15}$$

To ensure the rendered face fits appropriately in the image (neither too small nor too large), we define the initial distance from the camera to the origin of the world coordinate system as d. Note that the 3D face in world space is centered at the origin. Subsequently, we define the camera location as $T = [t_x + \Delta t_x, t_y + \Delta t_y, t_z + \Delta t_z]^T$, where (t_x, t_y, t_z) represents the initial camera location such that $d = \sqrt{t_x^2 + t_y^2 + t_z^2}$, and $(\Delta t_x, \Delta t_y, \Delta t_z)$ are offsets to be estimated during the 3D face reconstruction.

We allow the camera's FOV to be adjustable. To ensure the rendered face maintains a similar size in the image after a change in FOV, the updated initial distance from the camera to the origin is calculated as $d' = (f'/f)d$, where f' is the new focal length derived from the updated FOV θ' using Equation 5.15. Unlike existing image-based 3D face reconstruction methods that use a fixed camera intrinsic matrix, our approach allows the focal length to vary based on an estimated FOV, thereby enabling more accurate modeling of perspective distortion during reconstruction.

Synthetic Data Generation

PanoHead-FOV Dataset: The construction of PanoHead-FOV is based on two methods: the NeRF-StyleGAN-based 3D-consistent face image generation method, PanoHead [3], and the 3D Gaussian-based head avatar creation method, Gaussian Déjà-vu [219]. We first synthesize front-view face images of 10,000 randomly generated identities via PanoHead. Next, we employ Gaussian Déjà-vu to reconstruct the corresponding 3D Gaussian-based heads. Unlike PanoHead's implicit NeRF-based 3D representation, the explicit 3D Gaussian representation in Gaussian Déjà-vu enables us to render face images with extreme FOV settings without introducing unwanted artifacts.

We empirically define a discrete set of nine FOV values (in degrees), $\Phi = \{20, 25, 30, 35, 40, 45, 50, 55, 60\}$. For each identity, we render five images with

random yaw and pitch angles under each FOV in Φ. The choice of random yaw and pitch angles follows the findings described in [3, 28], where yaw and pitch are sampled from truncated normal distributions. Specifically, yaw angles are sampled with a standard deviation of 16° within the range $[-45^\circ, 45^\circ]$, and pitch angles are sampled with a standard deviation of 2° within the range $[-15^\circ, 15^\circ]$. Figure 5.22 shows some example face images rendered with different FOVs using our method. This figure illustrates how changes in FOV affect the perspective distortion and the appearance of the face in the image.

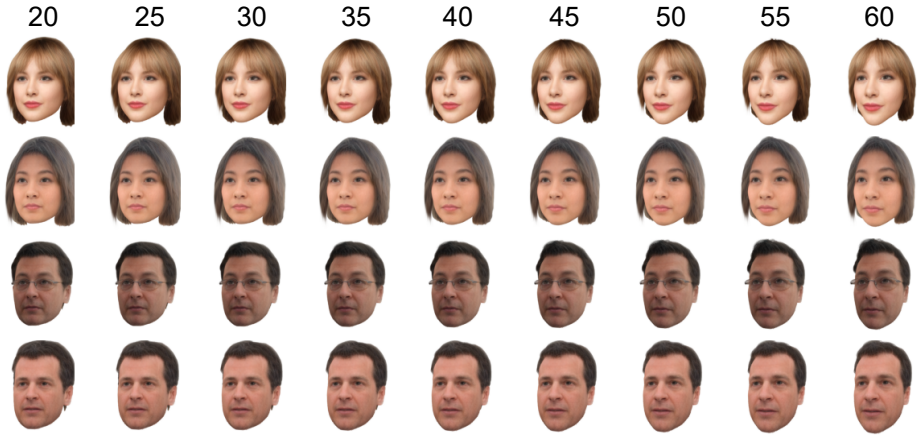


Figure 5.22: Example rendered face images. Each row presents images of the same 3D face rendered under varying FOVs.

The first 9,500 identities in the PanoHead-FOV dataset are used for training the FOV estimation network, while the remaining 500 identities are reserved for validation.

NeRSemble-FOV Dataset: We use 14 subjects from the NeRSemble dataset [93] to construct the NeRSemble-FOV dataset. For each subject in NeRSemble, there are 10 video sequences containing multi-view frames (16 camera views), with the subject displaying different facial expressions in each video. From each video sequence, we select 7 frames by sampling every 10th frame (including the first frame) from the first 60 frames. In the first frame of each video, the subject has neutral expression.

As the original NeRSemble dataset does not provide ground-truth 3D face

shapes, we use a 3D face (with bare head) modeling software, FaceBuilder, to manually model the neutral expression 3D face shapes using the first frame (multi-view) of each subject. Figure 5.23 shows our manually modeled 3D head meshes, with textures automatically generated by the software. From the modeled 3D head meshes, we extract the 3D locations of the facial region, following the approach described in [154]. The extracted facial region point clouds serve as the ground-truth 3D face shapes for the dataset to evaluate the 3D face reconstruction accuracy. Figure 5.24 shows the ground-truth face shape of a subject.



Figure 5.23: Showcase of our modeled 3D bare head meshes based on multi-view images.

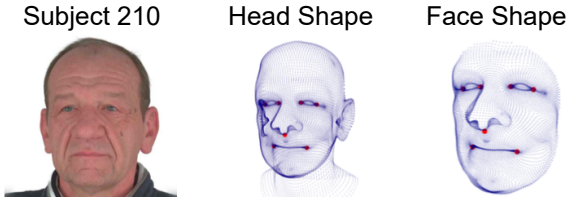


Figure 5.24: The image of subject 210, its head shape and the extracted face shape. Red Dots: 7 face landmarks [154].

We adapt the Gaussian Déjà-vu method to accurately reconstruct the 3D Gaussian-based head using the multi-view frames. For the reconstructed head from each selected frame, we render five images with random yaw and pitch angles for each FOV in Φ, following the same procedure as used in the construction of the PanoHead-FOV dataset. In total, our NeRSemble-FOV dataset comprises 44,100 rendered

face images. Figure 5.25 shows some example rendered face images.

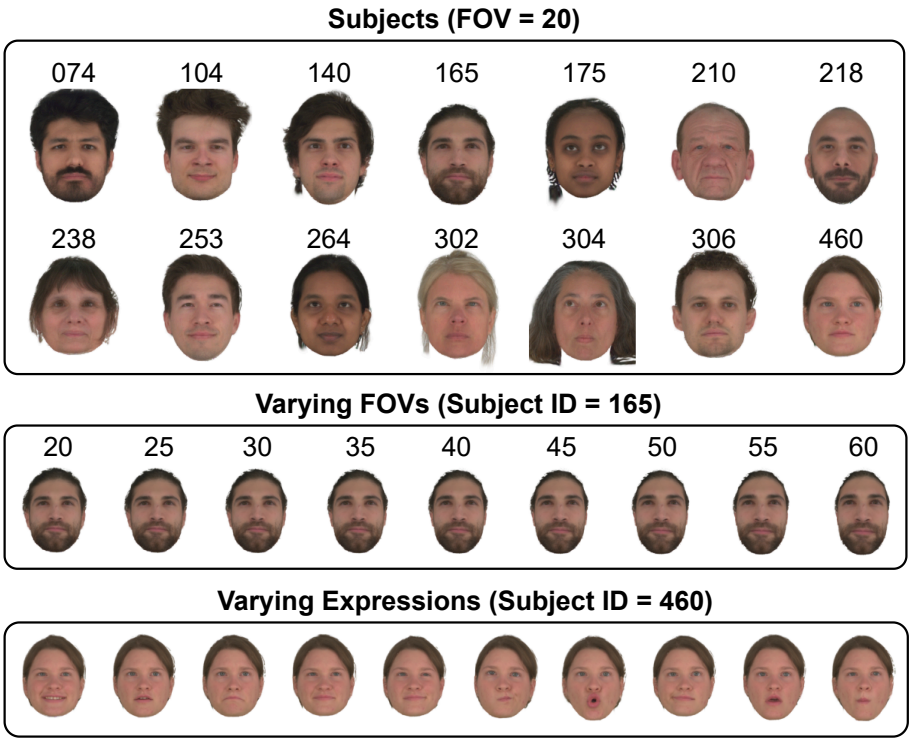


Figure 5.25: Example Images from the NeRSemble-FOV Dataset: The first block shows rendered images of 14 subjects (subject IDs above). The second block displays the same subject under varying FOVs, and the third shows varying expressions.

FOV-Net: FOV Estimation Network

We design a neural network, namely FOV-Net, to estimate the FOV of a given face image. FOV-Net is built on a pre-trained ArcFace network with a ResNet-50 backbone [37]. We add a fully connected layer after the last layer of ArcFace for FOV classification. The input to FOV-Net is a 112×112 image, and the output is a nine-dimensional vector. Prior to input, the face image is cropped and aligned according to the pre-processing standard described in [219]. Denote the output of FOV-Net after softmax as $p \in \mathbb{R}^9$, where p_i represents the estimated probability of

the i^{th} FOV label Φ_i, the estimated FOV in the continuous range can be computed as:

$$\hat{\theta} = \sum_{i=1}^{9} p_i \Phi_i \tag{5.16}$$

We train FOV-Net on our synthetic PanoHead-FOV dataset. Since FOV-Net is built on ArcFace, which was pre-trained on real face images, we freeze the pre-trained weights in the earlier layers to preserve the learned patterns on real data. To adapt FOV-Net to the synthetic dataset, we fine-tune the last convolutional block of ResNet-50 and the fully connected layers only. The training loss function is defined as:

$$\mathcal{L}_{fov} = \mathcal{L}_{ce} + \mathcal{L}_{mse}, \tag{5.17}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss for FOV classification, $\mathcal{L}_{mse}$ is the mean squared error between the ground-truth and the estimated FOVs.

3D Face Reconstruction with Estimated FOV

We employ the 3DMM-based head model namely FLAME [109] for 3D face reconstruction from 2D images. Our reconstruction approach is fitting-based, optimizing the FLAME coefficients and the 6DoF camera pose to ensure the projected face shape aligns with the face in the given 2D image. Our pre-trained FOV-Net is used to estimate the FOV for initializing the camera's focal length. To initialize the FLAME coefficients, we leverage a single-image-based reconstruction network, DECA [48]. We then refine these coefficients and the camera pose by minimizing the following facial landmarks alignment loss function:

$$\mathcal{L}_{lmk} = \lambda_{face}\mathcal{L}_{face} + \lambda_{jaw}\mathcal{L}_{jaw} + \lambda_{ear}\mathcal{L}_{ear}, \tag{5.18}$$

where $\mathcal{L}_{face}$, $\mathcal{L}_{jaw}$, and $\mathcal{L}_{ear}$ represent the L2 distances between the projected landmarks and the detected landmarks on the given 2D image, and λs are loss term weights. Specifically, $\mathcal{L}_{face}$ and $\mathcal{L}_{jaw}$ compute the losses for facial and jawline landmarks respectively [19]. Additionally, $\mathcal{L}_{ear}$ computes the loss for the ear landmarks. Figure 5.26 shows our reconstruction pipeline.

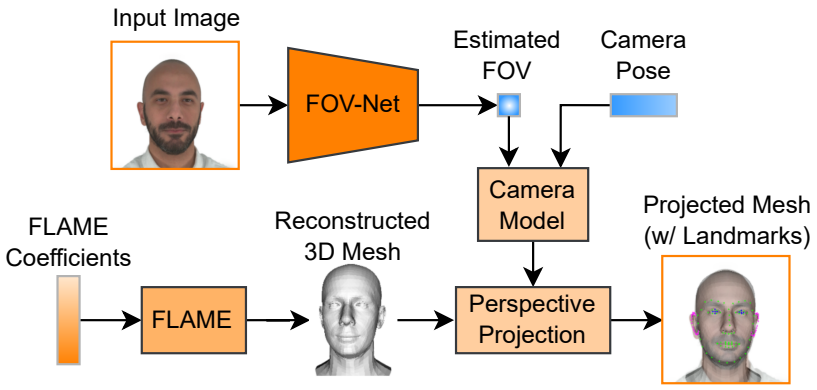


Figure 5.26: Our fitting-based 3D face reconstruction pipeline.

5.5.4 Experiments

Implementation Details

We use the Adam optimizer with a constant learning rate of 10^{-4} to optimize FOV-Net. The batch size is set to 32, and at each iteration, images are randomly loaded from the training set of PanoHead-FOV. The training process spans 50,000 iterations and takes approximately 3 hours on a single NVIDIA A6000 GPU.

In the fitting-based 3D face reconstruction process, we use Mediapipe's face landmark detector to detect facial and jawline landmarks. For ear landmark detection, we train a convolutional neural network following [22] on the i-Bug ear dataset [244]. We set the initial camera location to $(0, 0, 1)$, thus $d = 1$. The loss term weights λ_{face}, λ_{jaw} and λ_{ear} are set to 500, 100, and 100, respectively. We use the Adam optimizer for fitting, employing a two-stage optimization strategy. First, we optimize the 6DoF camera pose for 200 iterations with a learning rate of 0.01. Next, we optimize the FLAME coefficients for 800 iterations with a learning rate of 0.005.

FOV Estimation Performance

We run our FOV-Net on the NeRSemble-FOV dataset to estimate the camera FOV for each image, and evaluate the results separately on the first frames and the entire

dataset. The results are presented as confusion matrices in Figure 5.27. We also compute the Pearson correlations between the estimated and ground-truth FOV values, which are 0.837 for the first frames and 0.782 for the entire dataset. We attribute the higher FOV estimation accuracy on the first frames to their neutral expressions, whereas other frames contain more exaggerated expressions. Since FOV-Net was trained on a synthetic dataset with primarily subtle expressions, its performance is better on the neutral expressions than the exaggerated ones.

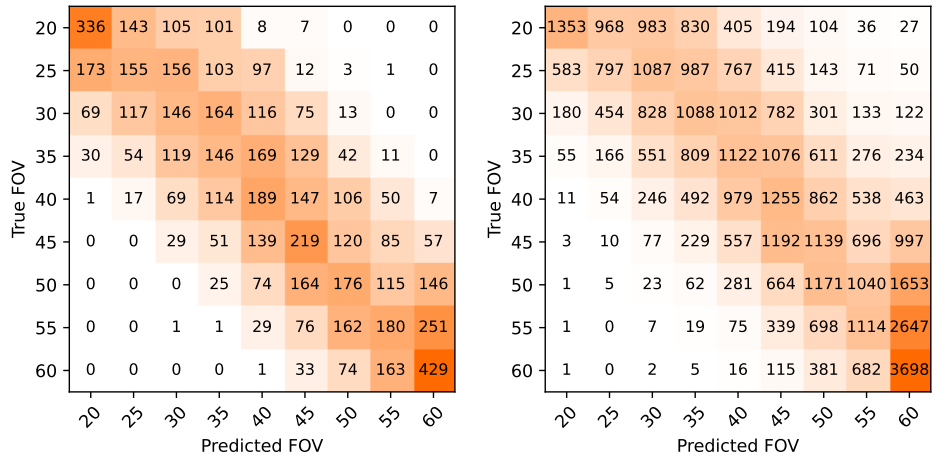


Figure 5.27: Confusion matrices of the testing FOV estimation results on NeRSemble-FOV dataset. Left: result on first frames (neutral expressions) only; Right: result on the entire dataset.

Accurate FOV estimation is crucial for consistent 3D face reconstruction, as perspective distortion affects alignment and accuracy. In the next section, we demonstrate how FOV-Net improves reconstruction consistency and accuracy under varying distortion levels in input images.

3D Face Reconstruction Experiment

Since we only have ground-truth face point clouds for neutral expressions, we evaluate image-based reconstruction using only the first frames in NeRSemble-FOV. We compare our method with neural network 3D face reconstruction approaches, including DECA [48], MICA [248], and 6DoF-Face [85]. In this experiment, we use all five random views under the same FOV to fit the FLAME coefficients. For

neural network-based methods, we reconstruct each view and average the results to obtain the final shape. To ensure comparability, the ground-truth face shapes are scaled to match the FLAME model's mean shape. The reconstructed 3D face mesh is then aligned and rescaled to the point cloud using the seven facial landmarks, following the procedure in [154]. Finally, we calculate the mean point-to-surface distance following [12] to quantify the difference between the ground-truth face point clouds and the reconstructed 3D face meshes. The quantitative results are shown in Table 5.11, demonstrating that our method provides more consistent reconstruction across varying distortion levels. Example reconstructed 3D meshes projected to 2D are shown in Figure 5.28. Notably, when the FOV is large (resulting in higher distortion), our method achieves better alignment.

Table 5.11: Mean point-to-surface distance $\downarrow$ (scaled by 10^{-3}) between ground-truth point clouds and reconstructed meshes. The values are averaged across all subjects.

Method	FOV 20	FOV 30	FOV 40	FOV 50	FOV 60
DECA [48]	1.929	1.934	1.973	2.045	2.107
MICA [248]	**1.903**	1.961	1.972	2.001	2.075
6DoF-Face [85]	3.620	3.529	3.387	3.219	3.074
Ours	1.915	**1.883**	**1.872**	**1.835**	**1.899**

Ablation Study

In our ablation study, we evaluate the consistency of the reconstruction of the same face rendered under different FOVs. We expect the reconstructed 3D faces from images of the same face with different FOVs to be similar. We use the reconstructed faces from images with FOV = 20 (low distortion) as references, and compute the mean per-vertex Euclidean distance between these references and those reconstructed from images with higher FOVs (FOV $>$ 20). We compare the use of our estimated FOV in fitting with using a fixed FOV approach. The fixed FOV is set to 20, consistent with existing 3D face reconstruction methods, which assume low perspective distortion. The results, presented in Table 5.12, demonstrate that incorporating the FOV estimated by FOV-Net significantly improves reconstruction consistency across varying FOVs. Additionally, Figure 5.29 provides a visual

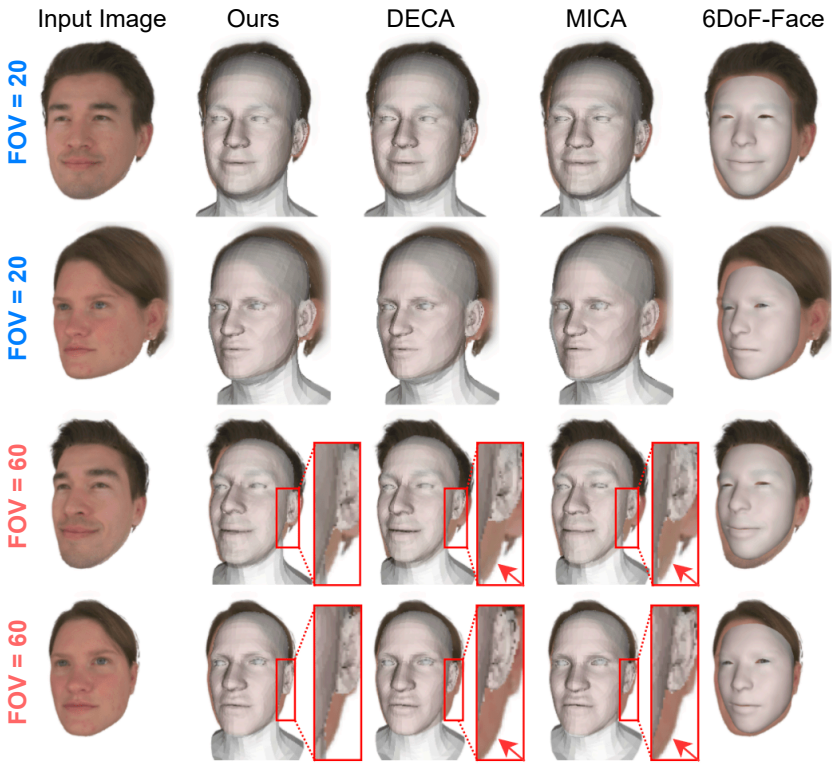


Figure 5.28: Qualitative reconstruction results. The arrows indicate regions with noticeable misalignment. FOV value indicates the image's ground-truth FOV.

example, showing that using a fixed FOV = 20 during fitting results in less consistent reconstructed 3D faces when the input image has larger perspective distortion (FOV $>$ 20). This highlights the advantage of adapting the FOV based on the input image.

Table 5.12: Mean vertex distance $\downarrow$ (scaled by 10^{-3}) between reconstructed 3D faces using images with FOV = 20 and reconstructed 3D faces using images with FOV $>$ 20.

Method	FOV 30	FOV 40	FOV 50	FOV 60
w/ Fixed FOV	0.673	1.014	1.357	1.670
w/ Estimated FOV	**0.545**	**0.754**	**0.958**	**1.143**

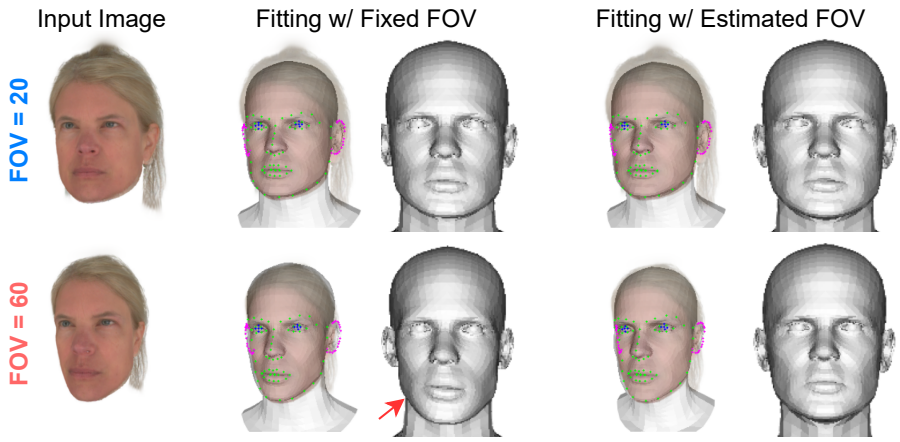


Figure 5.29: Ablation study results of the reconstructions on rendered images of the same face with two FOVs. For each method, we show the rendered mesh and projected landmarks aligned to image, and the rendered mesh in canonical camera pose. Arrow indicates visible distortion in reconstructed face.

5.5.5 Summary

A virtual camera is essential for 2D-to-3D face reconstruction, enabling accurate projection of the reconstructed 3D face onto the 2D image plane. In this section, we propose estimating the virtual camera's FOV to better handle perspective distortion in 2D face images, enabling more accurate 3D face reconstruction. We introduce two datasets to train our FOV estimation network, FOV-Net, and evaluate its impact on FOV estimation and 3D face reconstruction accuracy. Experiments show that estimating the FOV significantly improves reconstruction consistency across varying levels of perspective distortion. While the results are promising, further improvements are needed to enhance computational efficiency and robustness, especially for faces with exaggerated expressions or those deviating from the training data. Future work could focus on neural network models for 2D-to-3D face reconstruction that explicitly account for perspective distortion.

5.6 Chapter Summary

In this chapter, we primarily address the personalization problem in 3D head creation, aiming to produce highly realistic, animatable 3D head avatars that faithfully resemble the target individual. We use short monocular videos as training data to reduce reliance on costly multi-view capture setups. Our methods build on the 3DGS representation, whose explicit geometry enables controllable facial expression modeling by binding 3D Gaussians to a 3DMM-based head mesh. At the same time, its radiance-field formulation allows learning photorealistic appearance details from 2D images or videos. Our first method, Déjà-vu, reduces personalization learning cost by introducing a single-image–based 3D Gaussian head reconstruction model that provides a strong initialization for subsequent video-based refinement. Our second method, ArchitectHead, tackles the challenge of balancing rendering efficiency and visual fidelity by incorporating continuous level of detail control into 3DGS-based head avatars. Both methods achieve state-of-the-art realism and controllability in facial animation. Finally, we demonstrate an additional application of our 3DGS-based head avatar model by generating synthetic head images to train a camera field-of-view estimation network, which improves fitting-based 3D face reconstruction under perspective distortion.

Chapter 6

Conclusion and Discussion

This thesis investigates how to jointly achieve realism, controllability, and affordability in 3D head creation. Rather than committing to a single 3D representation, this thesis explores multiple 3D representations, each selected to address the limitations encountered at different stages of 3D head creation.

We begin by focusing on controllability through 3D mesh-based face shape manipulation (Chapter 3). This choice is motivated by two key considerations. First, the face is a critical component of the head that encodes a person's identity. Second, 3D meshes are a widely used and foundational representation in 3D head modeling, offering strong interpretability and intuitive control for shape editing.

Despite these advantages, mesh-based representations are inherently limited in their ability to capture photorealistic details. To overcome this limitation, Chapter 4 shifts the focus toward improving realism in rendered 3D face images. In particular, we build upon Neural Radiance Fields (NeRF) [128], which provide significantly greater representational capacity than meshes. Moreover, by enabling training from in-the-wild face images, this approach improves affordability in data collection.

However, NeRF-based methods rely on implicit representations and are computationally expensive to render, which limits their practicality in real-time applications such as animatable 3D head avatars. To address this efficiency bottleneck, Chapter 5 builds on recent advances in explicit 3D representations. Specifically, we leverage 3D Gaussian Splatting (3DGS) [90], an emerging 3D representation that

supports efficient rendering while retaining the ability to model complex geometry and realistic appearance. Using this representation, we further improve realism through personalization, while simultaneously achieving controllability in 3D head animation and affordability in both training data requirements and rendering efficiency.

Together, these three chapters form a progressive path toward jointly addressing realism, controllability, and affordability in 3D head creation. In the following section, we summarize the main contributions of the proposed methods.

6.1 Contribution

First, we study mesh-based controllable 3D face shape deformation and propose approaches that enable intuitive 3D face editing and cost-effective 3D face stylization (**Chapter 3**).

Our first method, NEO-3DF [216], is an editing-oriented 3D face creation framework built upon modeling the 3D face based on face parts, where the face is segmented into semantic parts (e.g., nose and eyes), and each part is modeled independently using a variational autoencoder (VAE). This part-based architecture offers greater local flexibility than existing global 3D face models, enabling more intuitive, localized face shape editing. In addition, the increased flexibility of individual facial parts leads to improved 3D-to-2D alignment accuracy in single-image-based 3D face reconstruction.

Our second method addresses 3D face shape stylization through a weakly-supervised learning framework [220]. We first formulate face stylization as a deformation transfer problem [171]. Our aim is to transfer facial features (such as nose shape and eye size) from a reconstructed 3D face to an artist-designed style template. Traditional deformation transfer methods require computationally expensive optimization for each input face. To address this problem, we propose a weakly-supervised learning approach with a novel mesh smoothness regularization loss to approximate the deformation transfer process. As a result, the trained network achieves stylization speeds that are approximately 3,000 times faster than the traditional deformation transfer method.

Despite the advantages of 3D meshes, such as their explicit surface structure,

compatibility with 2D texture maps, and efficient rendering, mesh-based representations inherently struggle to model highly detailed and flexible structures, such as hair and fine-scale appearances. Then, in **Chapter 4**, we study the use of an implicit 3D representation method, NeRF, to model the 3D face. NeRF enables the use of 2D images as training data to learn an implicit 3D representation via a neural network and allows the rendering of photorealistic images. This feature implies that there is no need to have 3D scans as training data, which lowers the data collection cost.

A significant challenge with NeRF-based 3D face models is controllability, as implicit 3D representations are less easy to manipulate than explicit 3D meshes. Existing works extend the idea of 3D Morphable Face Models (3DMMs) [14, 45] to NeRF-based 3D face modeling to achieve both photorealistic quality and improved controllability [52, 71, 247]. However, these methods do not fully explore the use of in-the-wild face images as training data to achieve better generalization and lower data collection costs.

Therefore, we propose StyleMorpheus, a novel NeRF-based 3D-aware morphable face model trained solely on in-the-wild face images [221]. StyleMorpheus utilizes a disentangled latent space that mimics the four parameter groups of a 3DMM (identity, expression, appearance, and lighting). The training of StyleMorpheus involves autoencoder-based training for single-image face reconstruction and Style-GAN-based training [87] to facilitate the learning of photorealistic details. As a result, StyleMorpheus achieves improved reconstruction quality compared to existing methods while also relying on less expensive in-the-wild images as training data.

Finally, building on our observations and foundations in Chapter 3 and Chapter 4, we extend our work to the creation of highly realistic and animatable 3D head avatars using only monocular videos. In **Chapter 5**, we present 3DGS-based head avatar creation methods, Déjà-vu [219] and ArchitectHead [218]. 3DGS relies on 3D Gaussian primitives to form a radiance field that allows learning flexible geometry and complex appearance. In our methods, the 3D Gaussian primitives are bound to a 3D mesh-based head model to achieve controllable facial animation. The rendering speed of 3DGS is also more efficient than that of NeRF-based methods, making it suitable for applications that require real-time performance and

efficiency, such as telepresence and virtual meetings.

In Déjà-vu, we advance existing 3DGS-based head avatar methods by proposing a reconstruction network that enables single-image-based 3DGS head reconstruction. This reconstruction network is trained on both in-the-wild images and synthetic images to achieve improved generalization ability. We use this reconstruction network to initialize the 3DGS head avatar and further optimize it using a short monocular video of an individual to create a realistic personalized head avatar. We also propose learnable 3D Gaussian blendmaps to better model varying facial expressions. As a result, our method achieves state-of-the-art quality as well as a faster personalization training speed than existing methods.

To enable 3DGS-based head avatars to achieve a balance between rendering efficiency and visual quality, we introduce the level-of-detail (LOD) concept in our proposed ArchitectHead method, the first 3DGS-based head avatar method that supports continuous LOD control. Enabling LOD control in 3DGS is challenging because even nearby 3D Gaussians can differ significantly in shape, orientation, and color, making naive Gaussian merging prone to detail loss and artifacts. ArchitectHead addresses this challenge by parameterizing the head avatar in a 2D UV space and learning multi-level feature maps that represent latent 3D Gaussian features. We use a lightweight neural network to decode the Gaussian attributes from the corresponding latent features. By resampling feature maps at different resolutions via interpolation, the number of 3D Gaussians can be directly controlled. Our approach achieves state-of-the-art quality even at the lowest LOD setting, while the transition of LOD is smooth and continuous.

Overall, the work in this thesis forms a coherent research progression driven by the goal of achieving controllable, realistic, and cost-effective 3D face and head modeling. Each stage addresses the limitations of the previous one, progressing from geometric control to photorealistic appearance and ultimately to real-time and realistic animation, thereby establishing a unified framework for practical 3D head creation.

6.2 Limitations

Our mesh-based 3D face editing and stylization methods are not fully automatic and require some manual work, which limits their scalability. For example, in our face editing method, we must manually define the semantic meaning of each face editing controller (e.g., a controller that adjusts nose height). To construct each controller, we also need to annotate the corresponding vertex regions on the 3D face mesh in order to compute the geometric measurements used for editing. In addition, our face stylization method still requires an artist-designed 3D mesh template for each style.

Despite improved disentanglement in the latent space, our NeRF-based StyleMorpheus method remains less directly controllable than explicit mesh-based representations, particularly for precise geometric editing. Moreover, facial edits are performed in the latent space, which is less intuitive than direct geometric manipulation.

Our 3DGS-based head avatar methods rely on accurate 3D face mesh reconstruction and tracking. Inaccurate 3D-to-2D alignment and/or camera pose estimation can lead to artifacts in the learned head avatar models. In addition, we do not model hair dynamics, which causes the hair to appear stiff during rapid head motion.

6.3 Ethics Discussions

An important ethical issue is that, as highly realistic 3D head models and avatars can be created from short videos or images, the lowered technical barriers raise concerns about privacy and misuse. For example, a head avatar may be misused for impersonation or misleading content, making it difficult for viewers to distinguish synthetic content from real videos or images of the individual it represents. Potential strategies to prevent privacy and misuse of a 3D head avatar are: (1) disclosure and transparency in the use of synthetic head avatars, where the use of synthetic head avatars is clearly indicated to inform viewers that the content is generated; (2) digital watermarking and detection techniques to help identify generated content; and (3) raising public awareness of the increasing realism of synthetic data.

Another ethical consideration involves data bias and data usage. Training on

in-the-wild images may introduce demographic and cultural biases present in publicly available datasets. In addition, in-the-wild data collected from public sources may raise concerns about unclear usage rights and licensing issues. Potential solutions include careful dataset curation with attention to usage rights and diversity considerations, as well as transparency in dataset sources and sample distributions.

6.4 Future Work

Our future work will mainly focus on improving the practicality of 3D head avatars. In particular, we discuss three potential directions. The first is to better balance generalization and personalization in head avatar creation. The second is to enable continuous learning to improve head avatars over time. The third is to leverage multi-modal input signals, such as audio and facial landmarks, to improve the robustness of head avatar animation. In the following, we will discuss each direction in more detail.

6.4.1 Balance Generalization and Personalization in 3D Head Avatar Creation

A fundamental objective in machine learning is generalization, where the goal is to extend learned patterns from training data to unseen data. In the context of 3D head creation, learning a generalizable head model is important because it provides a strong prior that helps ensure the created 3D head model appears human-like. We have studied generalization in 3D face and head modeling in this thesis and observed that generalization alone struggled to create a realistic head avatar that was indistinguishable from the specific individual it represents. Therefore, we believe that personalization is necessary to achieve the goal of learning a personalized head avatar model.

In our work, Déjà-vu, we explore using a generalizable reconstruction model to initialize the parameters of the head avatar, thereby providing a better starting point for personalization. However, our reconstruction model is trained on images with limited variations in facial expressions and therefore does not generalize well to unseen facial expressions. Recent works explore training more generalizable models to reconstruct animatable head avatars [66, 94, 180]. However, these methods

emphasize generalization and often result in blurry details.

Based on these observations, we believe that we still need a promising research direction to study how to balance generalization and personalization. In particular, a generalizable model is expected to ensure plausibility and reduce optimization difficulty during the personalization stage. At the same time, personalization is expected to adapt the model to a specific individual using their data (e.g., short videos or images) and focus on high-frequency appearance (such as skin texture and wrinkles) and expression-specific details.

6.4.2 Continuous Learning in 3D Head Avatar Personalization

To reduce data collection costs and improve usability, it is desirable for 3D head avatar creation to rely on easily accessible input data. While recent works have explored creating animatable head avatars from a single image or a small number of images, these approaches often struggle to capture individual-specific details, particularly subtle or extreme facial expressions that occur during speech or natural interaction [66, 94, 180]. As a result, using monocular video data for personalization remains a practical choice.

However, a single short video is typically insufficient to cover the full range of a person's facial expressions and appearance variations. One promising direction is to enable continuous learning, where users can incrementally improve their head avatar models by providing additional video data captured at different times. For example, in video conferencing, the use over time naturally generates diverse facial motion data that could be leveraged to progressively refine the avatar.

At the same time, continuous learning introduces new challenges. Videos captured at different times may exhibit variations in hairstyle, accessories, lighting conditions, etc. These variations can introduce inconsistencies into the learning process. An open problem is how to guide the model to focus on stable and identity-related patterns, such as facial geometry and characteristic expressions, while ignoring temporary or less relevant changes. Developing effective strategies for continuous personalization under such unconstrained conditions remains an important and largely unexplored research direction.

6.4.3 Using Multi-Modal Input Signals to Drive 3D Head Avatar Animation

Existing methods for animating 3D head avatars typically map a single driving signal (e.g., audio or video frames) to expression-related parameters of the avatar model, such as blendshape weights or expression coefficients [33, 62, 108]. However, in practice, relying on a single modality is prone to noise and information loss. For example, the audio in real-world scenarios may be corrupted by background noise, such as traffic or conversations from other people. Similarly, video-based signals can suffer from occlusions, such as face masks or the mouth region being covered by a hand or other objects, leading to missing or unreliable visual cues. In both cases, using a single driving signal can result in inaccurate facial animation.

To address this issue, we consider combining audio and video modalities and employing an uncertainty-aware fusion strategy to integrate their signals. When one modality is likely to be noisy or unreliable, the complementary modality can be leveraged to mitigate or correct the final driving signal. In this way, more robust and stable facial animation can be achieved.

Bibliography

[1] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten. A survey on deep learning advances on different 3d data representations. *arXiv preprint arXiv:1808.01462*, 2018. → page 20

[2] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM transactions on graphics (TOG)*, 22(3):587–594, 2003. → pages 44, 46

[3] S. An, H. Xu, Y. Shi, G. Song, U. Y. Ogras, and L. Luo. Panohead: Geometry-aware 3d full-head synthesis in 360deg. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20950–20959, 2023. → pages 126, 127, 137, 166, 167, 218

[4] P. Arsomngern, C. Long, S. Suwajanakorn, and S. Nutanong. Towards pointsets representation learning via self-supervised learning and set augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1201–1216, 2021. → page 75

[5] F. Aurenhammer and R. Klein. Voronoi diagrams. *Handbook of computational geometry*, 5(10):201–290, 2000. → page 68

[6] Y. Bai, Y. Fan, X. Wang, Y. Zhang, J. Sun, C. Yuan, and Y. Shan. High-fidelity facial avatar reconstruction from monocular video with generative priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4541–4551, 2023. → pages 3, 84

[7] Z. Bai, Z. Cui, X. Liu, and P. Tan. Riggable 3d face reconstruction via in-network optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6216–6225, 2021. → pages 37, 40

[8] Z. Bai, F. Tan, S. Fanello, R. Pandey, M. Dou, S. Liu, P. Tan, and Y. Zhang. Efficient 3d implicit head avatar with mesh-anchored hash table blendshapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1975–1984, 2024. → page 4

[9] F. Barthel, A. Beckmann, W. Morgenstern, A. Hilsmann, and P. Eisert. Gaussian splatting decoder for 3d-aware generative adversarial networks. In *Workshop of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. → page 128

[10] J. Basset, A. Boukhayma, S. Wuhrer, F. Multon, and E. Boyer. Neural human deformation transfer. In *2021 International Conference on 3D Vision (3DV)*, pages 545–554. IEEE, 2021. → pages 62, 63

[11] P. Bergeron and P. LaChapelle. Tony de peltrie. *Film, SIGGRAPH-85 Film Show, San Francisco*, 1985. → pages 2, 4

[12] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992. → page 173

[13] C. Bi, X. Liu, and Z. Liu. Nerf-ad: Neural radiance field with attention-based disentanglement for talking face synthesis. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3490–3494. IEEE, 2024. → page 81

[14] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999. → pages 3, 28, 29, 37, 40, 44, 55, 64, 82, 84, 179

[15] Y. Blau and T. Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6228–6237, 2018. → page 111

[16] J. Booth, A. Roussos, S. Zafeiriou, A. Ponniah, and D. Dunaway. A 3d morphable model learnt from 10,000 faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5543–5552, 2016. → page 29

[17] A. Boreskov and E. Shikin. *Computer Graphics: From Pixels to Programmable Graphics Hardware*. CRC Press, 2013. → page 19

[18] G. Bouritsas, S. Bokhnyak, S. Ploumpis, M. Bronstein, and S. Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7213–7222, 2019. → page 40

[19] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *Proceedings of the IEEE international conference on computer vision*, pages 1021–1030, 2017. → page 170

[20] A. Caliskan, B. Kicanaoglu, and H. Kim. Pav: Personalized head avatar from unstructured video collection. In *European Conference on Computer Vision*, 2024. → page 81

[21] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018. → pages 48, 92, 212

[22] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. → page 171

[23] H. Cha, I. Lee, and H. Joo. Perse: Personalized 3d generative avatars from a single portrait. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15953–15962, 2025. → page 36

[24] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021. → page 85

[25] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. → pages 85, 86, 127

[26] F.-J. Chang, A. T. Tran, T. Hassner, I. Masi, R. Nevatia, and G. Medioni. Expnet: Landmark-free, deep, 3d facial expressions. In *2018 13th IEEE*

International Conference on Automatic Face & Gesture Recognition (FG 2018), pages 122–129. IEEE, 2018. → page 45

[27] A. Chen, Z. Chen, G. Zhang, K. Mitchell, and J. Yu. Photo-realistic facial details synthesis from single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9429–9439, 2019. → page 55

[28] E. M. Chen, S. Holalkere, R. Yan, K. Zhang, and A. Davis. Ray conditioning: Trading photo-consistency for photo-realism in multi-view image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23242–23251, 2023. → page 167

[29] Y. Chen, L. Wang, Q. Li, H. Xiao, S. Zhang, H. Yao, and Y. Liu. Monogaussianavatar: Monocular gaussian point-based head avatar. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–9, 2024. → pages 124, 127, 128, 149

[30] Z. Chen, R. Jiang, B. Duke, H. Zhao, and P. Aarabi. Exploring gradient-based multi-directional controls in gans. In *European Conference on Computer Vision*, pages 104–119. Springer, 2022. → page 106

[31] J. Chokkattu. My Virtual Avatar No Longer Looks Terrible in the Apple Vision Pro — wired.com. https://www.wired.com/story/apples-vision-pro-personas-look-so-much-better/, 2025. [Accessed 29-09-2025]. → page 2

[32] X. Chu, Y. Li, A. Zeng, T. Yang, L. Lin, Y. Liu, and T. Harada. Gpavatar: Generalizable and precise head avatar from image (s). *ICLR*, 2024. → page 127

[33] C. Chung, I. Fedorov, M. Huang, A. Karmanov, D. Korobchenko, R. Ribera, Y. Seol, et al. Audio2face-3d: Audio-driven realistic facial animation for digital avatars. *arXiv preprint arXiv:2508.16401*, 2025. → page 184

[34] F. Cole, D. Belanger, D. Krishnan, A. Sarna, I. Mosseri, and W. T. Freeman. Synthesizing normalized faces from facial identity features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3703–3712, 2017. → pages 90, 92

[35] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145, 1984. → page 23

[36] E. David J. *Introduction to computer graphics*. Open Textbook Library, 2023. → page 19

[37] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019. → pages 88, 90, 92, 96, 169

[38] Y. Deng, J. Yang, S. Xu, D. Chen, Y. Jia, and X. Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. → pages 3, 4, 29, 37, 38, 39, 41, 45, 46, 54, 56, 70, 84, 88, 99, 126

[39] Y. Deng, J. Yang, D. Chen, F. Wen, and X. 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. → pages 37, 40

[40] Y. Deng, D. Wang, X. Ren, X. Chen, and B. Wang. Portrait4d: Learning one-shot 4d head avatar synthesis using synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7119–7130, 2024. → pages 85, 86

[41] H. Dhamo, Y. Nie, A. Moreau, J. Song, R. Shaw, Y. Zhou, and E. Pérez-Pellitero. Headgas: Real-time animatable head avatars via 3d gaussian splatting. In *European Conference on Computer Vision*, pages 459–476. Springer, 2024. → pages 124, 128, 131, 149

[42] X. Dongye, H. Guo, L. Luo, H. Jiang, Y. Bao, Z. Tian, and D. Weng. Lodavatar: Hierarchical embedding and adaptive levels of detail with gaussian splatting for enhanced human avatars. *arXiv preprint arXiv:2410.20789*, 2024. → pages 125, 151

[43] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988. → page 27

[44] T. Eames. https://www.imdb.com/news/ni57210806/, 2014. [Accessed 30-09-2025]. → page 7

[45] B. Egger, W. A. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)*, 39(5):1–38, 2020. → pages 3, 37, 40, 84, 179

[46] L. G. Farkas, J. C. Kolar, and I. R. Munro. Geography of the nose: a morphometric study. *Aesthetic plastic surgery*, 10(1):191–223, 1986. → pages 50, 216

[47] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 534–551, 2018. → pages 4, 29, 41

[48] Y. Feng, H. Feng, M. J. Black, and T. Bolkart. Learning an animatable detailed 3d face model from in-the-wild images. *ACM Transactions on Graphics (ToG)*, 40(4):1–13, 2021. → pages 3, 29, 38, 39, 84, 99, 108, 126, 170, 172, 173

[49] S. Foti, B. Koo, D. Stoyanov, and M. J. Clarkson. 3d shape variational autoencoder latent disentanglement via mini-batch feature swapping for bodies and faces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18730–18739, 2022. → pages 38, 44, 45, 56, 57

[50] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. → page 27

[51] G. Gafni, J. Thies, M. Zollhofer, and M. Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8649–8658, 2021. → page 148

[52] S. Galanakis, B. Gecer, A. Lattas, and S. Zafeiriou. 3dmm-rf: Convolutional radiance fields for 3d face modeling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3536–3547, 2023. → pages 83, 84, 85, 95, 99, 104, 105, 127, 139, 141, 179

[53] G. Gambetta. *Computer Graphics from Scratch: A Programmer's Introduction to 3D Rendering*. No Starch Press, 2021. → page 19

[54] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379*, 2022. → page 82

[55] L. Gao, J. Yang, Y.-L. Qiao, Y.-K. Lai, P. L. Rosin, W. Xu, and S. Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. → page 63

[56] P. Garrido, M. Zollhöfer, D. Casas, L. Valgaerts, K. Varanasi, P. Pérez, and C. Theobalt. Reconstruction of personalized 3d face rigs from monocular video. *ACM Transactions on Graphics (TOG)*, 35(3):1–15, 2016. → page 3

[57] B. Gecer, S. Ploumpis, I. Kotsia, and S. Zafeiriou. Ganfit: Generative adversarial network fitting for high fidelity 3d face reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1155–1164, 2019. → page 4

[58] D. Gerogiannis, F. P. Papantoniou, R. A. Potamias, A. Lattas, and S. Zafeiriou. Arc2avatar: Generating expressive 3d avatars from a single image via id guidance. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10770–10782, 2025. → pages 36, 128

[59] D. Ghafourzadeh, S. Fallahdoust, C. Rahgoshay, A. Beauchamp, A. Aubame, T. Popa, and E. Paquette. Local control editing paradigms for part-based 3d face morphable models. *Computer Animation and Virtual Worlds*, 32(6):e2028, 2021. → pages 38, 44, 45, 46, 50, 56

[60] P. Ghosh. Motion capture tech from Avatar films used in disease research — bbc.com. https://www.bbc.com/news/science-environment-64326125, 2023. [Accessed 03-10-2025]. → pages 2, 4

[61] S. Giebenhain, T. Kirschstein, M. Rünz, L. Agapito, and M. Nießner. Npga: Neural parametric gaussian avatars. *arXiv preprint arXiv:2405.19331*, 2024. → pages 124, 128, 153

[62] S. Gong, H. Li, J. Tang, D. Hu, S. Huang, H. Chen, T. Chen, and Z. Liu. Monocular and generalizable gaussian talking head animation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5523–5534, 2025. → page 184

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. → pages 9, 30, 34, 85, 101

[64] J. Gu, L. Liu, P. Wang, and C. Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *arXiv preprint arXiv:2110.08985*, 2021. → pages 85, 97, 115, 118, 119

[65] J. Gu, L. Liu, P. Wang, and C. Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=iUuzzTMUw9K. → pages 82, 127

[66] C. Guo, Z. Su, J. Wang, S. Li, X. Chang, Z. Li, Y. Zhao, G. Wang, and R. Huang. Sega: Drivable 3d gaussian head avatar from a single image. *arXiv preprint arXiv:2504.14373*, 2025. → pages 182, 183

[67] F. Han, S. Ye, M. He, M. Chai, and J. Liao. Exemplar-based 3d portrait stylization. *IEEE Transactions on Visualization and Computer Graphics*, 29(2):1371–1383, 2021. → pages 38, 41, 62, 63

[68] Y. He, X. Gu, X. Ye, C. Xu, Z. Zhao, Y. Dong, W. Yuan, Z. Dong, and L. Bo. Lam: Large avatar model for one-shot animatable gaussian head. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–13, 2025. → page 128

[69] T. Hempel, A. A. Abdelrahman, and A. Al-Hamadi. 6d rotation representation for unconstrained head pose estimation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2496–2500. IEEE, 2022. → page 103

[70] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. → pages xviii, 9, 30, 35

[71] Y. Hong, B. Peng, H. Xiao, L. Liu, and J. Zhang. Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20374–20384, 2022. → pages 4, 14, 82, 83, 84, 85, 91, 95, 97, 101, 105, 119, 121, 127, 139, 141, 179

[72] H. Hoppe. Progressive meshes. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 111–120. Association for Computing Machinery, 2023. → page 125

[73] M. Hu, G. Brazil, N. Li, L. Ren, and X. Liu. Camera self-calibration using human faces. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–8. IEEE, 2023. → page 165

[74] R. Hu, X. Wang, Y. Yan, and C. Zhao. Tgavatar: Reconstructing 3d gaussian avatars with transformer-based tri-plane. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025. → page 153

[75] Q. Huang, X. Huang, B. Sun, Z. Zhang, J. Jiang, and C. Bajaj. Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5815–5825, 2021. → pages 39, 67, 70

[76] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017. → page 85

[77] P. J. Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer, 1992. → page 155

[78] W. Jang, Y. Jung, H. Kim, G. Ju, C. Son, J. Son, and S. Lee. Toonify3d: Stylegan-based 3d stylized face generator. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. → page 41

[79] J. Jiang, C. Liang, J. Yang, G. Lin, T. Zhong, and Y. Zheng. Loopy: Taming audio-driven portrait avatar with long-term motion dependency. *arXiv preprint arXiv:2409.02634*, 2024. → page 36

[80] K. Jiang, S.-Y. Chen, F.-L. Liu, H. Fu, and L. Gao. Nerffaceediting: Disentangled face editing in neural radiance fields. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. → page 81

[81] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016. → page 88

[82] J. Johnson, N. Ravi, J. Reizenstein, D. Novotny, S. Tulsiani, C. Lassner, and S. Branson. Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 courses*, pages 1–1. ACM, 2020. → pages 23, 24

[83] B. Kabadayi, W. Zielonka, B. L. Bhatnagar, G. Pons-Moll, and J. Thies. Gan-avatar: Controllable personalized gan-based human head avatar. In *2024 International Conference on 3D Vision (3DV)*, pages 882–892. IEEE, 2024. → page 127

[84] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984. → pages 25, 86, 98

[85] Y. Kao, B. Pan, M. Xu, J. Lyu, X. Zhu, Y. Chang, X. Li, and Z. Lei. Toward 3d face reconstruction in perspective projection: Estimating 6dof face pose from monocular image. *IEEE Transactions on Image Processing*, 32: 3080–3091, 2023. → pages 126, 165, 172, 173

[86] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. → pages 85, 99, 102

[87] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. → pages 70, 85, 91, 103, 107, 115, 126, 138, 139, 179, 218

[88] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. → pages 14, 54, 83, 85, 97, 102, 104, 127

[89] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021. → pages 85, 97, 127

[90] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. → pages 15, 20, 26, 27, 123, 127, 177

[91] M. J. Kesterke, Z. D. Raffensperger, C. L. Heike, M. L. Cunningham, J. T. Hecht, C. H. Kau, N. L. Nidey, L. M. Moreno, G. L. Wehby, M. L. Marazita, et al. Using the 3d facial norms database to investigate craniofacial sexual dimorphism in healthy children, adolescents, and adults. *Biology of sex differences*, 7(1):1–14, 2016. → pages 50, 216

[92] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. → pages 9, 30, 31

[93] T. Kirschstein, S. Qian, S. Giebenhain, T. Walter, and M. Nießner. Nersemble: Multi-view radiance field reconstruction of human heads. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023. → pages 3, 126, 167

[94] T. Kirschstein, J. Romero, A. Sevastopolsky, M. Nießner, and S. Saito. Avat3r: Large animatable gaussian reconstruction model for high-fidelity 3d head avatars. *arXiv preprint arXiv:2502.20220*, 2025. → pages 128, 182, 183

[95] T. Koizumi and W. A. Smith. Shape from semantic segmentation via the geometric rényi divergence. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2312–2321, 2021. → pages 29, 37, 38, 41, 46, 49, 82

[96] J. Kulhanek, M.-J. Rakotosaona, F. Manhardt, C. Tsalicoglou, M. Niemeyer, T. Sattler, S. Peng, and F. Tombari. Lodge: Level-of-detail large-scale gaussian splatting with efficient rendering. *arXiv preprint arXiv:2505.23158*, 2025. → page 151

[97] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)*, 39(6):1–14, 2020. → page 24

[98] Y. Lan, F. Tan, D. Qiu, Q. Xu, K. Genova, Z. Huang, S. Fanello, R. Pandey, T. Funkhouser, C. C. Loy, et al. Gaussian3diff: 3d gaussian diffusion for 3d full head synthesis and editing. *arXiv preprint arXiv:2312.03763*, 2023. → pages 124, 128

[99] A. Lattas, S. Moschoglou, B. Gecer, S. Ploumpis, V. Triantafyllou, A. Ghosh, and S. Zafeiriou. Avatarme: Realistically renderable 3d facial reconstruction" in-the-wild". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 760–769, 2020. → page 55

[100] B. H. Le and Z. Deng. Interactive cage generation for mesh deformation. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 1–9, 2017. → page 43

[101] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. → page 111

[102] C.-H. Lee, Z. Liu, L. Wu, and P. Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. → pages 54, 104, 107, 139

[103] G.-H. Lee and S.-W. Lee. Uncertainty-aware mesh decoder for high fidelity 3d face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6100–6109, 2020. → page 39

[104] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. H. Pighin, and Z. Deng. Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)*, 1(8):2, 2014. → pages 43, 84

[105] D. Li, K. Zhao, W. Wang, Y. Ma, B. Peng, Y. Zhang, and J. Dong. S 3 d-nerf: Single-shot speech-driven neural radiance field for high fidelity talking head synthesis. In *European Conference on Computer Vision*, pages 365–382. Springer, 2024. → page 81

[106] H. Li, C. Chen, T. Shi, Y. Qiu, S. An, G. Chen, and X. Han. Spherehead: stable 3d full-head synthesis with spherical tri-plane representation. In *European Conference on Computer Vision*, pages 324–341. Springer, 2024. → page 85

[107] J. Li, C. Cao, G. Schwartz, R. Khirodkar, C. Richardt, T. Simon, Y. Sheikh, and S. Saito. Uravatar: Universal relightable gaussian codec avatars. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. → page 128

[108] L. Li, Y. Li, Y. Weng, Y. Zheng, and K. Zhou. Rgbavatar: Reduced gaussian blendshapes for online modeling of head avatars. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10747–10757, 2025. → pages 125, 127, 149, 153, 155, 156, 158, 184, 225, 226

[109] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4d scans. *ACM Transactions on Graphics (TOG)*, 36(6):1–17, 2017. → pages 1, 4, 28, 29, 45, 84, 130, 132, 150, 151, 170

[110] X. Li, J. Wang, Y. Xuan, G. Yao, and Y. Pan. Ggavatar: Geometric adjustment of gaussian head avatar. *arXiv preprint arXiv:2405.11993*, 2024. → pages 4, 124, 128

[111] J. Lin, Y. Yuan, T. Shao, and K. Zhou. Towards high-fidelity 3d face reconstruction from in-the-wild images using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5891–5900, 2020. → pages 40, 45, 55

[112] Z. Lin, J. Lin, L. Li, Y. Yuan, and Z. Zou. High-quality 3d face reconstruction with affine convolutional networks. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2495–2503, 2022. → page 99

[113] F. Liu, Q. Zhao, X. Liu, and D. Zeng. Joint face alignment and 3d face reconstruction with application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):664–678, 2018. → page 37

[114] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019. → pages 23, 24

[115] S. Liu, T. Li, W. Chen, and H. Li. A general differentiable mesh renderer for image-based 3d reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. → page 23

[116] X. Liu, Z. Liu, and C. Bi. Nerf-3dtalker: Neural radiance field with 3d prior aided audio disentanglement for talking head synthesis. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025. → page 81

[117] Y. Liu, C. Luo, L. Fan, N. Wang, J. Peng, and Z. Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. In *European Conference on Computer Vision*, pages 265–282. Springer, 2024. → page 151

[118] J. Luo, J. Liu, and J. Davis. Splatface: Gaussian splat face reconstruction leveraging an optimizable surface. *arXiv preprint arXiv:2403.18784*, 2024. → pages 124, 128

[119] B. Ma, Y. Cong, and J. Dong. Topology-aware graph convolution network for few-shot incremental 3-d object learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023. → page 67

[120] S. Ma, Y. Weng, T. Shao, and K. Zhou. 3d gaussian blendshapes for head avatar animation. *arXiv preprint arXiv:2404.19398*, 2024. → pages 124, 128, 131

[121] Z. Ma, X. Zhu, G.-J. Qi, Z. Lei, and L. Zhang. Otavatar: One-shot talking face avatar with controllable tri-plane rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16901–16910, 2023. → page 127

[122] J. Martinez, E. Kim, J. Romero, T. Bagautdinov, S. Saito, S.-I. Yu, S. Anderson, M. Zollhöfer, T.-L. Wang, S. Bai, et al. Codec avatar studio: Paired human captures for complete, driveable, and generalizable avatars. *Advances in Neural Information Processing Systems*, 37:83008–83023, 2024. → pages 3, 4

[123] T. Martyniuk, O. Kupyn, Y. Kurlyak, I. Krashenyi, J. Matas, and V. Sharmanska. Dad-3dheads: A large-scale dense, accurate and diverse dataset for 3d head alignment from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20942–20952, 2022. → page 38

[124] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. → pages 86, 98

[125] Y. Men, H. Liu, Y. Yao, M. Cui, X. Xie, and Z. Lian. 3dtoonify: Creating your high-fidelity 3d stylized avatar easily from 2d portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10127–10137, 2024. → page 41

[126] M. Merras, S. El Hazzat, A. Saaidi, K. Satori, and A. G. Nazih. 3d face reconstruction using images from cameras with varying parameters. *International Journal of Automation and Computing*, 14:661–671, 2017. → page 165

[127] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. → page 102

[128] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. → pages 21, 25, 27, 82, 88, 98, 126, 177

[129] N. Milef, D. Seyb, T. Keeler, T. Nguyen-Phuoc, A. Božič, S. Kondguli, and C. Marshall. Learning fast 3d gaussian splatting rendering using continuous level of detail. In *Computer Graphics Forum*, page e70069. Wiley Online Library, 2025. → page 151

[130] H. Mohaghegh, F. Boussaid, H. Laga, H. Rahmani, and M. Bennamoun. Robust monocular 3d face reconstruction under challenging viewing conditions. *Neurocomputing*, 520:82–93, 2023. → page 84

[131] M. Mori, K. F. MacDorman, and N. Kageki. The uncanny valley [from the field]. *IEEE Robotics & automation magazine*, 19(2):98–100, 2012. → pages 5, 6

[132] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, 2006. → pages 39, 67

[133] M. Olano, K. Akeley, J. C. Hart, W. Heidrich, M. McCool, J. L. Mitchell, and R. Rost. Real-time shading. In *ACM SIGGRAPH 2004 Course Notes*, pages 1–es. ACM New York, NY, USA, 2004. → page 118

[134] H. Onizuka, D. Thomas, H. Uchiyama, and R.-i. Taniguchi. Landmark-guided deformation transfer of template facial expressions for automatic generation of avatar blendshapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. → pages xviii, 13, 62, 65, 71, 72, 73, 78

[135] Y. Pan, S. Tan, S. Cheng, Q. Lin, Z. Zeng, and K. Mitchell. Expressive talking avatars. *IEEE Transactions on Visualization and Computer Graphics*, 2024. → pages 39, 41

[136] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. → pages 21, 33

[137] F. I. Parke. Computer generated animation of faces. In *Proceedings of the ACM annual conference-Volume 1*, pages 451–457, 1972. → pages 2, 4

[138] F. I. Parke. *A parametric model for human faces.* The University of Utah, 1974. → pages 2, 6

[139] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 sixth IEEE international conference on advanced video and signal based surveillance*, pages 296–301. Ieee, 2009. → pages 28, 29, 45, 46, 70, 84

[140] J. C. Pérez, T. Nguyen-Phuoc, C. Cao, A. Sanakoyeu, T. Simon, P. Arbeláez, B. Ghanem, A. Thabet, and A. Pumarola. Styleavatar: Stylizing animatable head avatars. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8678–8687, 2024. → page 41

[141] J. Piao, K. Sun, Q. Wang, K.-Y. Lin, and H. Li. Inverting generative adversarial renderer for face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15628, 2021. → page 37

[142] S. Ploumpis, E. Ververas, E. O'Sullivan, S. Moschoglou, H. Wang, N. Pears, W. A. Smith, B. Gecer, and S. Zafeiriou. Towards a complete 3d morphable model of the human head. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4142–4160, 2020. → page 29

[143] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. → page 36

[144] S. Qian, T. Kirschstein, L. Schoneveld, D. Davoli, S. Giebenhain, and M. Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20299–20309, 2024. → page 4

[145] S. Qian, T. Kirschstein, L. Schoneveld, D. Davoli, S. Giebenhain, and M. Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20299–20309, 2024. → pages 4, 15, 124, 125, 127, 128, 140, 143, 151, 156, 158, 225, 226

[146] P. Raina, F. Taubner, M. Tuli, E. W. Teh, and K. Ferreira. Prismavatar: Real-time animated 3d neural head avatars on edge devices. *arXiv preprint arXiv:2502.07030*, 2025. → page 81

[147] N. Ramanathan and R. Chellappa. Modeling age progression in young faces. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 387–394. IEEE, 2006. → pages 50, 216

[148] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018. → page 40

[149] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. → page 151

[150] S. C. Rhee, K.-S. Woo, and B. Kwon. Biometric study of eyelid shape and dimensions of different races with references to beauty. *Aesthetic plastic surgery*, 36(5):1236–1245, 2012. → pages 50, 216

[151] R. A. Roberts, R. K. dos Anjos, A. Maejima, and K. Anjyo. Deformation transfer survey. *Computers & Graphics*, 94:52–61, 2021. → page 62

[152] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. → page 36

[153] S. Saito, G. Schwartz, T. Simon, J. Li, and G. Nam. Relightable gaussian codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2024. → pages 4, 124, 128

[154] S. Sanyal, T. Bolkart, H. Feng, and M. Black. Learning to regress 3D face shape and expression from an image without 3D supervision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7763–7772, June 2019. → pages 168, 173

[155] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. → pages 45, 48, 92, 212

[156] Y. Seo, Y. S. Choi, H. S. Son, and Y. Uh. Flod: Integrating flexible level of detail into 3d gaussian splatting for customizable rendering. *arXiv preprint arXiv:2408.12894*, 2024. → pages 125, 151

[157] S. Serengil and A. Ozpinar. A benchmark of facial recognition pipelines and co-usability performances of modules. *Journal of Information Technologies*, 17(2):95–107, 2024. doi:10.17671/gazibtd.1399077. URL https://dergipark.org.tr/en/pub/gazibtd/issue/84331/1399077. → pages 137, 218

[158] G. Serifi and M. C. Bühler. Hypergaussians: High-dimensional gaussian splatting for high-fidelity animatable face avatars. *arXiv preprint arXiv:2507.02803*, 2025. → page 153

[159] Z. Shao, Z. Wang, Z. Li, D. Wang, X. Lin, Y. Zhang, M. Fan, and Z. Wang. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, pages 1606–1616, 2024. → pages 124, 125, 127, 128

[160] J. Shen, Y. Qian, and X. Zhan. Lod-gs: Achieving levels of detail using scalable gaussian soup. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 671–680, 2025. → page 151

[161] T. Shi, Y. Yuan, C. Fan, Z. Zou, Z. Shi, and Y. Liu. Face-to-parameter translation for game character auto-creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 161–170, 2019. → page 44

[162] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. → page 99

[163] Z. Shi, S. Peng, Y. Xu, A. Geiger, Y. Liao, and Y. Shen. Deep generative models on 3d representations: A survey. *arXiv preprint arXiv:2210.15663*, 2022. → page 29

[164] Y. Shih, W.-S. Lai, and C.-K. Liang. Distortion-free wide-angle portraits on camera phones. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. → page 126

[165] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. → page 99

[166] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. → pages 85, 98

[167] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, 2002. → pages 29, 84

[168] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015. → page 35

[169] L. Song, Y. Zhou, Z. Xu, Y. Zhou, D. Aneja, and C. Xu. Streamme: Simplify 3d gaussian avatar within live stream. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–10, 2025. → page 128

[170] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007. → pages 12, 39, 43, 49, 52, 67

[171] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004. → pages 13, 62, 64, 65, 66, 178

[172] J. Sun, X. Wang, Y. Zhang, X. Li, Q. Zhang, Y. Liu, and J. Wang. Fenerf: Face editing in neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7672–7682, 2022. → pages 5, 82, 85, 86

[173] J. Sun, X. Wang, L. Wang, X. Li, Y. Zhang, H. Zhang, and Y. Liu. Next3d: Generative neural texture rasterization for 3d-aware head avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20991–21002, 2023. → pages 4, 86, 118, 119, 127

[174] K. Sun, S. Wu, Z. Huang, N. Zhang, Q. Wang, and H. Li. Controllable 3d face synthesis with conditional generative occupancy fields. *Advances in Neural Information Processing Systems*, 35:16331–16343, 2022. → pages 85, 86

[175] Z. Sun, Z. Peng, Y. Ma, Y. Chen, Z. Zhou, Z. Zhou, G. Zhang, Y. Zhang, Y. Zhou, Q. Lu, and Y.-J. Liu. Streamavatar: Streaming diffusion models for real-time interactive human avatars. 2025. URL https://arxiv.org/abs/2512.22065. → page 36

[176] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. → pages 48, 212

[177] R. Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022. → page 19

[178] S. Tan and M. L. Mayrovouniotis. Reducing data dimensionality through optimizing neural network inputs. *AIChE Journal*, 41(6):1471–1480, 1995. → page 33

[179] J. Tang, D. Davoli, T. Kirschstein, L. Schoneveld, and M. Niessner. Gaf: Gaussian avatar reconstruction from monocular videos via multi-view diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5546–5558, 2025. → page 128

[180] F. Taubner, R. Zhang, M. Tuli, and D. B. Lindell. Cap4d: Creating animatable 4d portrait avatars with morphable multi-view diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5318–5330. IEEE Computer Society, 2025. → pages 36, 128, 182, 183

[181] J. R. Tena, F. De la Torre, and I. Matthews. Interactive region-based linear 3d face models. In *ACM SIGGRAPH 2011 papers*, pages 1–10. ACM, 2011. → page 38

[182] K. Teotia, X. Pan, H. Kim, P. Garrido, M. Elgharib, and C. Theobalt. Hq3davatar: High quality implicit 3d head avatar. *ACM Transactions on Graphics*, 2024. → page 127

[183] D. Terzopoulos and K. Waters. Physically-based facial modelling, analysis, and animation. *The journal of visualization and computer animation*, 1(2): 73–80, 1990. → page 2

[184] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library, 2022. → page 82

[185] J. Thies, M. Zollhöfer, and M. Nießner. Deferred neural rendering: Image synthesis using neural textures. *Acm Transactions on Graphics (TOG)*, 38 (4):1–12, 2019. → page 86

[186] L. Tran and X. Liu. Nonlinear 3d face morphable model. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7346–7355, 2018. → pages 4, 29, 37, 40, 41, 57, 82, 88, 96, 101

[187] Unknown. MetaHuman Creator — metahuman.unrealengine.com. https://metahuman.unrealengine.com/, 2025. [Accessed 03-10-2025]. → page 4

[188] Unknown. Furious 7 | Wētā FX — wetafx.co.nz. https://www.wetafx.co.nz/films/filmography/furious-7, 2025. [Accessed 29-09-2025]. → page 2

[189] Unknown. Google Beam - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Google_Beam, 2025. [Accessed 29-09-2025]. → page 2

[190] Unknown. Metaverse - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Metaverse, 2025. [Accessed 29-09-2025]. → page 2

[191] T. Vetter and V. Blanz. Estimating coloured 3d face models from single images: An example based approach. In *European conference on computer vision*, pages 499–513. Springer, 1998. → pages 2, 40, 84

[192] C. Wang, J. Huang, R. Zhang, Q. Wang, H. Yang, P. Wan, H. Huang, C. Ma, and W. Xu. Physically based facial texture generation in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. → page 4

[193] C. Wang, H.-Y. Peng, Y.-T. Liu, J. Gu, and S.-M. Hu. Diffusion models for 3d generation: A survey. *Computational Visual Media*, 11(1):1–28, 2025. → pages 30, 36

[194] D. Wang, H. Meng, Z. Cai, Z. Shao, Q. Liu, L. Wang, M. Fan, X. Zhan, and Z. Wang. Headevolver: Text to head avatars via expressive and attribute-preserving mesh deformation. *arXiv preprint arXiv:2403.09326*, 2024. → page 41

[195] H. Wang, F. Tan, Z. Bai, Y. Zhang, S. Liu, Q. Xu, M. Chai, A. Prabhu, R. Pandey, S. Fanello, et al. Lightavatar: Efficient head avatar as dynamic neural light field. In *European Conference on Computer Vision*, pages 183–201. Springer, 2024. → page 127

[196] J. Wang, X. Li, J. Xie, F. Xu, and H. Gao. Gaussianhead: Impressive 3d gaussian-based head avatars with dynamic hybrid neural field. *arXiv e-prints*, pages arXiv–2312, 2023. → pages 124, 128

[197] J. Wang, Y. Qiu, K. Chen, Y. Ding, and Y. Pan. Fully automatic blendshape generation for stylized characters. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 347–355. IEEE, 2023. → pages 39, 41, 62, 63

[198] J. Wang, J.-C. Xie, X. Li, F. Xu, C.-M. Pun, and H. Gao. Gaussianhead: High-fidelity head avatars with learnable gaussian derivation. *IEEE Transactions on Visualization and Computer Graphics*, 2025. → page 128

[199] S. Wang, T. Simon, I. Santesteban, T. Bagautdinov, J. Li, V. Agrawal, F. Prada, S.-I. Yu, P. Nalbone, M. Gramlich, et al. Relightable full-body gaussian codec avatars. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–12, 2025. → pages 2, 7

[200] Z. Wang. 3d representation methods: A survey. *arXiv preprint arXiv:2410.06475*, 2024. → page 20

[201] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. → page 135

[202] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. → page 139

[203] A. Watercutter. 'The Irishman' Gets De-Aging Right—No Tracking Dots Necessary — wired.com. https://www.wired.com/story/the-irishman-netflix-ilm-de-aging/, 2019. [Accessed 29-09-2025]. → page 2

[204] K. Waters. A muscle model for animation three-dimensional facial expression. *Acm siggraph computer graphics*, 21(4):17–24, 1987. → page 6

[205] F. Windisch, L. Radl, T. Köhler, M. Steiner, D. Schmalstieg, and M. Steinberger. A lod of gaussians: Unified training and rendering for ultra-large scale reconstruction with external memory. *arXiv preprint arXiv:2507.01110*, 2025. → page 151

[206] E. Wood, T. Baltrusaitis, C. Hewitt, S. Dziadzio, T. J. Cashman, and J. Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3681–3691, 2021. → pages 37, 40

[207] F. Wu, L. Bao, Y. Chen, Y. Ling, Y. Song, S. Li, K. N. Ngan, and W. Liu. Mvf-net: Multi-view 3d face morphable model regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 959–968, 2019. → page 3

[208] Y. Wu, H. Xu, X. Tang, X. Chen, S. Tang, Z. Zhang, C. Li, and X. Jin. Portrait3d: Text-guided high-quality 3d portrait generation using pyramid representation and gans prior. *ACM Transactions on Graphics (TOG)*, 43 (4):1–12, 2024. → page 41

[209] W. Xia and J.-H. Xue. A survey on deep generative 3d-aware image synthesis. *ACM Computing Surveys*, 56(4):1–34, 2023. → page 29

[210] J. Xiang, X. Gao, Y. Guo, and J. Zhang. Flashavatar: High-fidelity digital avatar rendering at 300fps. *arXiv preprint arXiv:2312.02214*, 2023. → pages 4, 15, 124, 125, 128, 130, 140, 143, 149, 150, 152, 154, 156, 158, 225, 226

[211] Y. Xiao, H. Zhu, H. Yang, Z. Diao, X. Lu, and X. Cao. Detailed facial geometry recovery from multi-view images by learning an implicit function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2839–2847, 2022. → page 3

[212] L. Xie, X. Wang, H. Zhang, C. Dong, and Y. Shan. Vfhq: A high-quality dataset and benchmark for video face super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 657–666, 2022. → page 120

[213] Z. Xing, Q. Feng, H. Chen, Q. Dai, H. Hu, H. Xu, Z. Wu, and Y.-G. Jiang. A survey on video diffusion models. *ACM Computing Surveys*, 57(2):1–42, 2024. → page 36

[214] H. Xu, G. Song, Z. Jiang, J. Zhang, Y. Shi, J. Liu, W. Ma, J. Feng, and L. Luo. Omniavatar: Geometry-guided controllable 3d head synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12814–12824, 2023. → pages 85, 86, 127

[215] Y. Xu, B. Chen, Z. Li, H. Zhang, L. Wang, Z. Zheng, and Y. Liu. Gaussian head avatar: Ultra high-fidelity head avatar via dynamic gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1931–1941, 2024. → pages 15, 124, 128

[216] P. Yan, J. Gregson, Q. Tang, R. Ward, Z. Xu, and S. Du. Neo-3df: Novel editing-oriented 3d face creation and reconstruction. In *Proceedings of the Asian Conference on Computer Vision*, pages 486–502, 2022. → pages 4, 5, 76, 77, 82, 84, 178

[217] P. Yan, R. Ward, D. Wang, Q. Tang, and D. Shan. Learning disentangled features for nerf-based face reconstruction. In *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2023. → page 99

[218] P. Yan, R. Ward, Q. Tang, and S. Du. Architecthead: Continuous level of detail control for 3d gaussian head avatars. *arXiv preprint arXiv:2510.05488*, 2025. → page 179

[219] P. Yan, R. Ward, Q. Tang, and S. Du. Gaussian déjà-vu: Creating controllable 3d gaussian head-avatars with enhanced generalization and personalization abilities. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 276–286. IEEE, 2025. → pages 4, 126, 128, 156, 158, 166, 169, 179, 225, 226

[220] P. Yan, R. K. Ward, Q. Tang, and S. Du. Neural 3d face shape stylization based on single style template via weakly supervised learning. *IEEE Transactions on Visualization and Computer Graphics*, 2025. → page 178

[221] P. Yan, R. K. Ward, D. Wang, Q. Tang, and S. Du. Stylemorpheus: Learning a stylegan-based 3d-aware morphable face model with a disentangled style space. *Neurocomputing*, page 131329, 2025. → page 179

[222] L. Yang, G. Zoss, P. Chandran, M. Gross, B. Solenthaler, E. Sifakis, and D. Bradley. Learning a generalized physical face model from data. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024. → page 7

[223] Z. Yang, B. Gong, and K. Chen. Lod-gs: Level-of-detail-sensitive 3d gaussian splatting for detail conserved anti-aliasing. *arXiv preprint arXiv:2507.00554*, 2025. → page 151

[224] V. Ye, R. Li, J. Kerr, M. Turkulainen, B. Yi, Z. Pan, O. Seiskari, J. Ye, J. Hu, M. Tancik, and A. Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025. → page 155

[225] Z. Ye, M. Xia, Y. Sun, R. Yi, M. Yu, J. Zhang, Y.-K. Lai, and Y.-J. Liu. 3d-carigan: an end-to-end solution to 3d caricature generation from normal face photos. *IEEE Transactions on Visualization and Computer Graphics*, 29(4):2203–2210, 2021. → pages 38, 41

[226] W. Yifan, F. Serena, S. Wu, C. Öztireli, and O. Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019. → page 28

[227] K. Yun, C. Kim, H. Shin, and J. Noh. Ffacenerf: Few-shot face editing in neural radiance fields. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10825–10835, 2025. → pages 81, 85, 86

[228] D. Zhang, Y. Liu, L. Lin, Y. Zhu, K. Chen, M. Qin, Y. Li, and H. Wang. Hravatar: High-quality and relightable gaussian head avatar. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26285–26296, 2025. → page 125

[229] J. Zhang, K. Chen, and J. Zheng. Facial expression retargeting from human to avatar made easy. *IEEE Transactions on Visualization and Computer Graphics*, 28(2):1274–1287, 2020. → pages 38, 39, 41, 62, 63

[230] J. Zhang, L. Lin, J. Zhu, and S. C. Hoi. Weakly-supervised multi-face 3d reconstruction. *arXiv preprint arXiv:2101.02000*, 2021. → page 165

[231] J. Zhang, Y. Lan, S. Yang, F. Hong, Q. Wang, C. K. Yeo, Z. Liu, and C. C. Loy. Deformtoon3d: Deformable neural radiance fields for 3d toonification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9144–9154, 2023. → page 41

[232] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023. → page 36

[233] R. Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pages 7324–7334. PMLR, 2019. → page 99

[234] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. → pages 99, 107, 135, 139, 154

[235] T. Zhang, X. Chu, Y. Liu, L. Lin, Z. Yang, Z. Xu, C. Cao, F. Yu, C. Zhou, C. Yuan, et al. Accurate 3d face reconstruction with facial component tokens. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9033–9042, 2023. → pages 3, 4, 126

[236] Y. Zhang, R. Su, J. Yu, and R. Li. 3d facial modeling, animation, and rendering for digital humans: A survey. *Neurocomputing*, 598:128168, 2024. → page 84

[237] M. Zheng, H. Zhang, H. Yang, and D. Huang. Neuface: Realistic 3d neural face rendering from multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16868–16877, 2023. → page 4

[238] X. Zheng, C. Wen, Z. Li, W. Zhang, Z. Su, X. Chang, Y. Zhao, Z. Lv, X. Zhang, Y. Zhang, et al. Headgap: Few-shot 3d head avatar via generalizable gaussian priors. In *2025 International Conference on 3D Vision (3DV)*, pages 946–957. IEEE, 2025. → pages 127, 128

[239] Y. Zheng, W. Yifan, G. Wetzstein, M. J. Black, and O. Hilliges. Pointavatar: Deformable point-based head avatars from videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21057–21067, 2023. → pages 127, 138, 155, 225

[240] J. Zhou, X. Tong, Z. Liu, and B. Guo. 3d cartoon face generation by local deformation mapping. *The Visual Computer*, 32:717–727, 2016. → pages 62, 63

[241] J. Zhou, H.-T. Wu, Z. Liu, X. Tong, and B. Guo. 3d cartoon face rigging from sparse examples. *The Visual Computer*, 34:1177–1187, 2018. → pages 39, 41

[242] P. Zhou, L. Xie, B. Ni, and Q. Tian. Cips-3d: A 3d-aware generator of gans based on conditionally-independent pixel synthesis. *arXiv preprint arXiv:2110.09788*, 2021. → page 85

[243] P. Zhou, L. Xie, B. Ni, and Q. Tian. Cips-3d++: End-to-end real-time high-resolution 3d-aware gans for gan inversion and stylization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. → page 85

[244] Y. Zhou and S. Zaferiou. Deformable models of ears in-the-wild for alignment and recognition. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 626–633. IEEE, 2017. → page 171

[245] H. Zhu, H. Yang, L. Guo, Y. Zhang, Y. Wang, M. Huang, M. Wu, Q. Shen, R. Yang, and X. Cao. Facescape: 3d facial dataset and benchmark for single-view 3d face reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. → page 37

[246] W. Zhu, H. Wu, Z. Chen, N. Vesdapunt, and B. Wang. Reda: reinforced differentiable attribute for 3d face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4958–4967, 2020. → pages 29, 37, 38, 40, 51, 82, 84

[247] Y. Zhuang, H. Zhu, X. Sun, and X. Cao. Mofanerf: Morphable facial neural radiance field. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 268–285. Springer, 2022. → pages 14, 83, 84, 95, 105, 118, 119, 127, 139, 141, 179

[248] W. Zielonka, T. Bolkart, and J. Thies. Towards metrical reconstruction of human faces. In *European conference on computer vision*, pages 250–269. Springer, 2022. → pages 3, 126, 172, 173

[249] W. Zielonka, T. Bolkart, and J. Thies. Instant volumetric head avatars. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4574–4584, 2023. → pages 155, 226

[250] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt. State of the art on monocular 3d face reconstruction, tracking, and applications. In *Computer graphics forum*, volume 37, pages 523–550. Wiley Online Library, 2018. → pages 3, 37

Appendices

Appendix A: NEO-3DF

A.1 NEO-3DF Network Architectures

The face image encoder in our framework is a FaceNet [155]. The network architecture is Inception-v1 [176], trained on the VGGFace2 dataset [21]. The encoded face representation vector (output of the FaceNet) is 512-dimensional. Other neural networks (E_i, F_i, D_i, and the offset regressor network) are Multilayer Perceptron (MLP) networks. Please see Tables A.1 to A.4 for network architectures of E_i, F_i, D_i, and the offset regressor network. $|V_i|$ is the number of vertices of segment i, and d_{z_i} is the dimension of latent representation of segment i. Note that the output of the offset regressor network has 10 dimensions because only five segments need offset along the y-axis (up and down) and the z-axis (forward and backward).

Table A.1: Shape Encoder Network (E_i) Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch, \|V_i\| \times 3]$
fc-1	input	ReLU	$[batch, \|V_i\| \times 3]$	$[batch, 128]$
fc-2	fc-1	ReLU	$[batch, 128]$	$[batch, 64]$
fc-mu	fc-2	Linear	$[batch, 64]$	$[batch, d_{z_i}]$
fc-sigma	fc-2	Exponential	$[batch, 64]$	$[batch, d_{z_i}]$

Table A.2: Latent Regressor Network (F_i) Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch, 512]$
fc-1	input	ReLU	$[batch, 512]$	$[batch, 128]$
fc-2	fc-1	ReLU	$[batch, 128]$	$[batch, 64]$
fc-mu	fc-2	Linear	$[batch, 64]$	$[batch, d_{z_i}]$
fc-sigma	fc-2	Exponential	$[batch, 64]$	$[batch, d_{z_i}]$

Table A.3: Shape Decoder Network (D_i) Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch, d_{z_i}]$
fc-1	input	ReLU	$[batch, d_{z_i}]$	$[batch, 128]$
fc-2	fc-1	ReLU	$[batch, 128]$	$[batch, 512]$
fc-output	fc-2	Linear	$[batch, 512]$	$[batch, \|V_i\| \times 3]$

Table A.4: Offset Regressor Network Architecture

Layer	Previous Layer	Activation	Input Size	Output Size
input	N/A	N/A	N/A	$[batch, d_{z_0}]$
fc-1	input	ReLU	$[batch, d_{z_0}]$	$[batch, 32]$
fc-2	fc-1	ReLU	$[batch, 32]$	$[batch, 32]$
fc-output	fc-2	Linear	$[batch, 32]$	$[batch, 10]$

A.2 Estimate Right-Hand-Side in ARAP

The right-hand-side estimation algorithm follows the standard ARAP formulation while ensuring differentiability (see Algorithm 2). The inputs are the original mesh vertices P, the vertices of the deformed mesh P', and the mesh Laplacian matrix L. Given the inputs, the algorithm first computes the Laplacian-weighted 1-ring edge vectors for both undeformed and deformed meshes. These edge vectors are then used to estimate a local rotation for each vertex. For each vertex, the covariance matrix between its undeformed and deformed neighborhoods is constructed and decomposed via SVD, from which a proper rotation matrix is extracted. Using these per-vertex rotations, each undeformed differential coordinate is rotated by the averaged rotations of its incident vertices to obtain the rotated differential coordinates. Finally, multiplying the result by L produces the ARAP right-hand-side term, which is then used in the global linear system to update vertex positions in each ARAP iteration. This procedure is repeated for a fixed number of iterations to optimize P'.

Algorithm 2 Estimate Right-Hand-Side

Input: P, P', L
Output: $estimate_rhs()$
Prepare Γ according to the mesh topology.
// Step 1: Get the un-deformed and deformed 1-ring vertex positions relative to each vertex, weighted by the Laplace edge weight.
$\Omega \leftarrow \mathbf{0}^{n_p \times n_p \times 3}$ // $\Omega \in \mathbb{R}^{n_p \times n_p \times 3}$ is a sparse matrix.
$\Omega' \leftarrow \mathbf{0}^{n_p \times n_p \times 3}$ // $\Omega' \in \mathbb{R}^{n_p \times n_p \times 3}$ is a sparse matrix.
for $i \in \{1..n_p\}$ **do**
  for $j \in N(i)$ **do**
    $(\Omega_{ijm})_{m \in \{1..3\}} \leftarrow (P_{im} - P_{jm}) \cdot L_{ij}$
    $(\Omega'_{ijm})_{m \in \{1..3\}} \leftarrow (P'_{im} - P'_{jm}) \cdot L_{ij}$
  end for
end for
// Step 2: Estimate the rotation from the un-deformed to the deformed mesh using the singular value decomposition (SVD).
$\mathbf{R} \leftarrow \mathbf{0}^{n_p \times 3 \times 3}$ // Initialize the rotations.
for $i \in \{1..n_p\}$ **do**
  $B \leftarrow \mathbf{0}^{|N(i)| \times 3}$
  $B' \leftarrow \mathbf{0}^{|N(i)| \times 3}$
  for $k \in \{1..|N(i)|\}$ **do**
    $j \leftarrow N(i)_k$
    $(B_{km})_{m \in \{1..3\}} \leftarrow \Omega_{ijm}$
    $(B'_{km})_{m \in \{1..3\}} \leftarrow \Omega'_{ijm}$
  end for
  $K \leftarrow B'^{\mathrm{T}} B$ // Compute the covariance matrix.
  $\mathbf{U}, \mathbf{S}, \mathbf{V}^{\mathrm{T}} \leftarrow SVD(K)$ // Take the Singular Value Decomposition (SVD) of K.
  $\mathbf{s} \leftarrow sgn(det(\mathbf{U}\mathbf{V}^{\mathrm{T}}))$ // $sgn(\cdot)$ is the sign function.
  $\mathbf{S}' \leftarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathbf{s} \end{bmatrix}$ // To ensure $det(\mathbf{R}_i) > 0$.
  $(\mathbf{R}_{imn})_{m,n \in \{1..3\}} \leftarrow (\mathbf{U}\mathbf{S}'\mathbf{V}^{\mathrm{T}})_{mn}$
end for
// Step 3: Rotate each un-deformed neighborhood edge by the average of its connected vertices to get the rotated differential coordinates for each vertex.
$\mathbf{D} \leftarrow \mathbf{0}^{n_p \times 3}$
for $i \in \{1..n_p\}$ **do**
  for $k \in \{1..|N(i)|\}$ **do**
    $j \leftarrow N(i)_k$
    $\mathbf{J} \leftarrow \mathbf{R}_j + \mathbf{R}_i$ // $\mathbf{J} \in \mathbb{R}^{3 \times 3}$.
    $(\mathbf{D}_{im})_{m \in \{1..3\}} \leftarrow \mathbf{D}_{im} + \sum_{n \in \{1..3\}} [\mathbf{J}_{mn} \times \Omega_{ikn} \times (0.5 \times L_{ij})]$
  end for
end for
Return: $L\mathbf{D}$

A.3 List of Selected Facial Features for Editing

We provide controllers of selected facial features for editing (see Table A.5). The features are measured according to the definitions in the literature (see the reference column).

Table A.5: List of facial features selected for editing.

Group	Feature Name	Editing Controller	Reference
Eyebrows	Curvature Strength	**EB-CS**	[150]
	Length	**EB-L**	
	Thickness	**EB-T**	
Eyes	Canthus Distance	**EY-CD**	[147, 150]
	Lateral Canthus (up/down)	**EY-LC**	
	Height	**EY-H**	
	Pupils Distance	**EY-PD**	
Nose	Bridge Width	**NS-BW**	[46, 91]
	Height	**NS-H**	
	Width	**NS-W**	
	Tip Depth	**NS-TD**	
	Tip Height	**NS-TH**	
	Tip Size	**NS-TS**	
Upper Lip	Height	**UL-H**	[91]
	Width	**UL-W**	
	Labial Fissure Width	**UL-LFW**	
Lower Lip	Height	**LL-H**	[91]
	Width	**LL-W**	
	End Height	**LL-EH**	
Rest	Facial Height	**RS-FH**	[91]
	Facial Width	**RS-FW**	
	Lower Facial Depth	**RS-LFD**	
	Lower Facial Height	**RS-LFH**	
	Mandibular Width	**RS-MW**	
	Middle Facial Depth	**RS-MFD**	
	Upper Facial Depth	**RS-UFD**	
	Upper Facial Height	**RS-UFH**	

We adjust each controller separately to be -3σ and $+3\sigma$ and show the change of each vertex position (Euclidean distance measured in millimeters) from the orig-

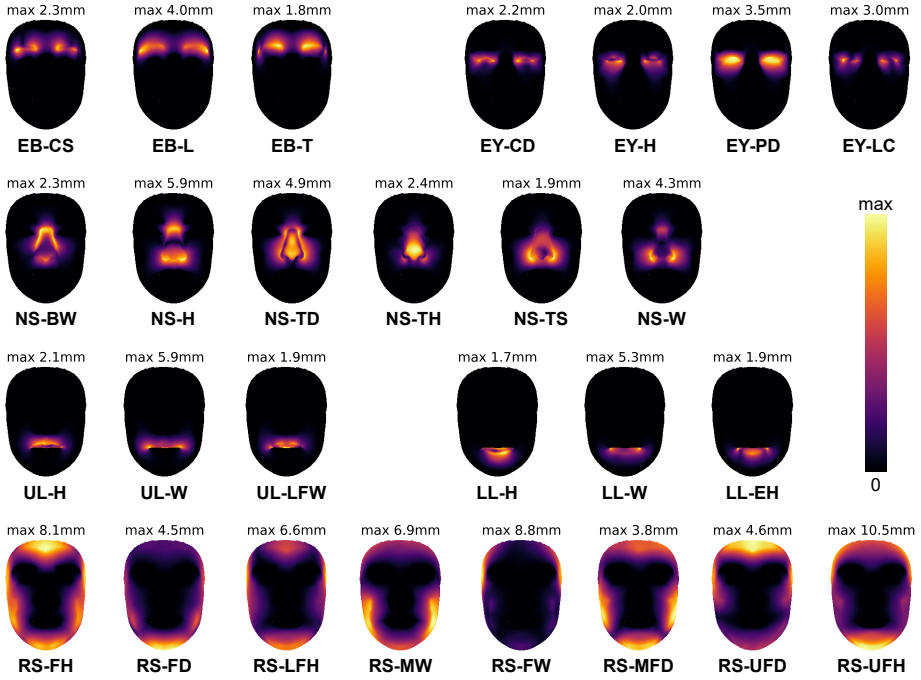


Figure A.1: Heatmaps of editing controllers. Controller names are defined in Appendix A.3.

inal mean face shape. The results are visualized as rendered 2D heatmaps in Figure A.1.

Appendix B: Déjà-vu

B.1 UV Gaussian Map in Déjà-vu Framework

We develop an automatic method to generate the interior of the mouth for the FLAME mesh. We first select the contour of the vertices around the mouth inside the FLAME mesh and split it to the upper line and lower line. Then we add an end line that defines the end of the mouth interior. We interpolate the vertices to create a dense funnel-shaped structure that can roughly represent the interior of the mouth. Figure B.2 illustrates the approach. Note that we only need to manually select the contour of vertices once, since all the FLAME reconstructed head meshes share the same topology.

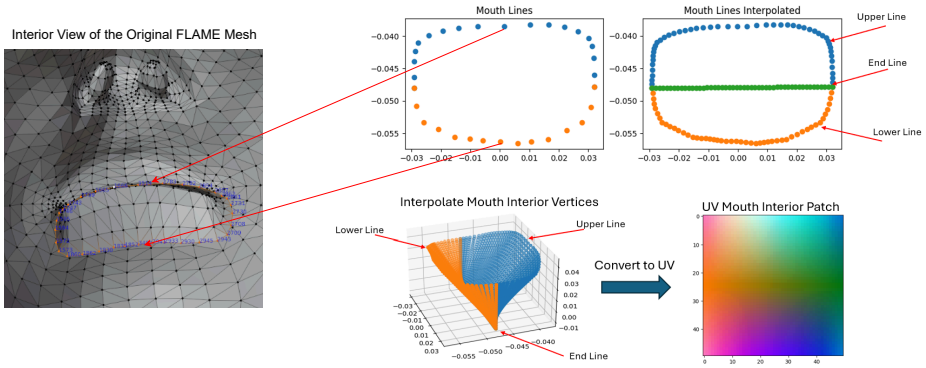


Figure B.2: Our method for automatically creating the mouth interior.

B.2 Details of Datasets Used in Déjà-vu Framework

We generate a synthetic dataset using the 3D-aware face image generation model, PanoHead [3]. Our synthetic dataset includes images of 18,000 synthesized identities, each rendered across 25 predefined camera views (see Figure B.3 for an example).

We utilize the DeepFace PyTorch Toolkit [157] to estimate age, gender, and emotion statistics on our synthetic dataset, as shown in Figure B.4.

The FFHQ dataset comprises 70,000 in-the-wild face images [87]. The reason



Figure B.3: An example identity generated from PanoHead, with 25 rendered views.

for using the FFHQ dataset is its wide coverage of faces from various ethnicities, genders, and age groups. However, the original FFHQ dataset primarily focused on the facial region; therefore, the facial region occupies a larger portion of the aligned image. This usually causes the head not to be fully visible in the image (e.g., part of the hair is outside the cropped image). We first collect the raw images (without alignment) used by FFHQ and realign the faces using our standard to cover the entire head (see Figure B.5).

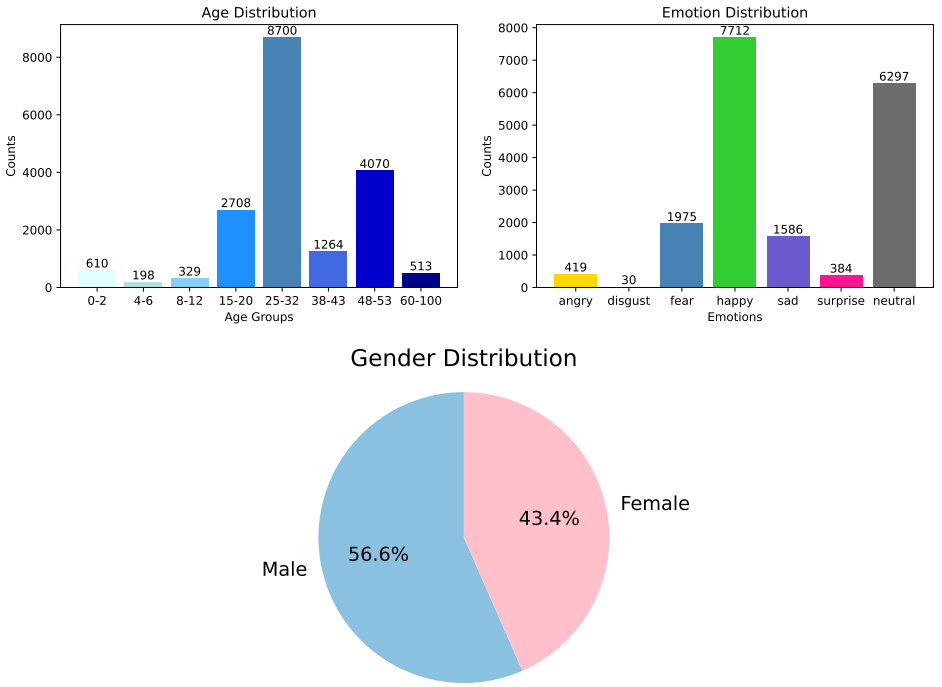


Figure B.4: Age, gender, and emotion distributions evaluated on the synthetic dataset.



Figure B.5: We realign the raw FFHQ images to cover the entire head.

B.3 Comparison of Déjà-vu and HeadNeRF in Novel View Synthesis

We compare our method with HeadNeRF to showcase the superior 3D view consistency of our 3D Gaussian-based head representation. The results are shown in Figure B.6. As shown, our method consistently outperforms HeadNeRF in maintaining view consistency, which is attributable to the explicit 3D structure represented by the 3D Gaussians. Notably, at extreme viewing angles, where HeadNeRF's renderings exhibit significant deterioration, our 3D Gaussian-based head reconstruction model continues to produce high-quality visual outputs.

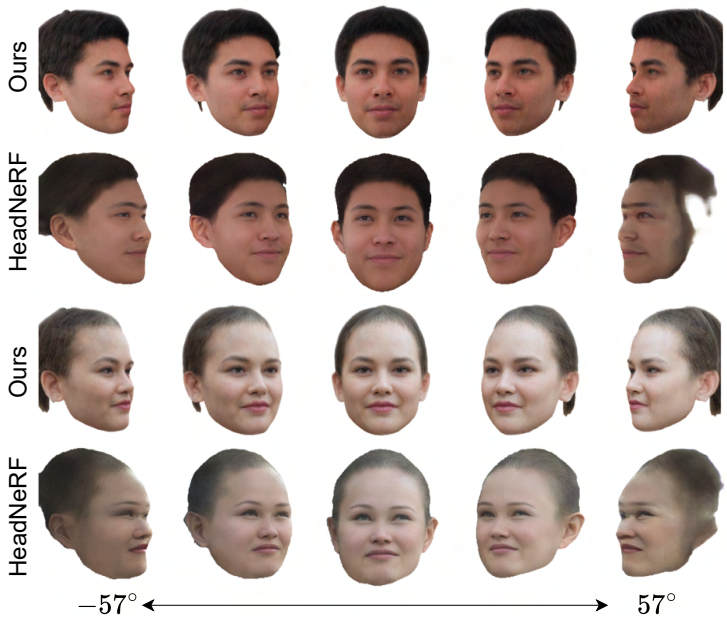


Figure B.6: Qualitative comparison with HeadNeRF across varying viewing angles. Our method works even at extreme angles.

Appendix C: ArchitectHead

C.1 ArchitectHead Framework Details

Positional Encoding

Let the input be

$$\mu \in \mathbb{R}^3,$$

where μ is the input 3D Gaussian location.

Define L as the number of frequency bands, we have the following frequencies:

$$\omega_i = 2^i, \quad i = 0, 1, \ldots, L-1.$$

For each frequency ω_i, we compute

$$\gamma_{\omega_i}(\mu) = \big[\sin(\omega_i \mu),\ \cos(\omega_i \mu)\big].$$

Then, we stack across all L frequency bands (original input included):

$$\gamma(\mu) = \big[\mu, \gamma_{\omega_0}(\mu), \gamma_{\omega_1}(\mu), \ldots, \gamma_{\omega_{L-1}}(\mu)\big].$$

The output dimension is

$$D_{pos} = 3 \cdot \Big(1 + 2L\Big).$$

We use 12 frequency bands, therefore the dimension after positional encoding is 75.

Mapping Network

The mapping network consists of three hidden layers, each with 64 neurons, followed by an output layer with K neurons. All hidden layers use *ReLU* activation, while the output layer has linear activation.

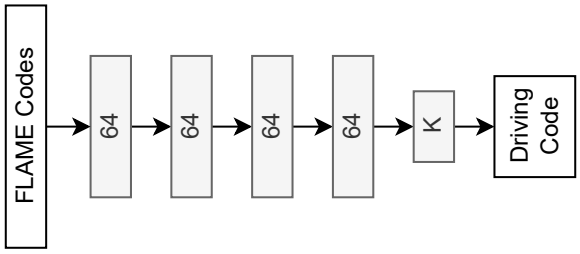


Figure C.7: Architecture of the mapping network $\mathcal{M}$.

Decoder Network

Our decoder $\mathcal{D}$ consists of five sub-networks, each sharing the same architecture except for the input dimension, output dimension, and the activation function of the output layer. Each sub-network contains four hidden layers with 256 neurons each, followed by an output layer with D_{attr} neurons corresponding to the target Gaussian attribute (one of $\mu \in \mathbb{R}^3, \mathbf{s} \in \mathbb{R}^3, \mathbf{q} \in \mathbb{R}^4, \alpha \in \mathbb{R}^1$, and $\mathbf{c} \in \mathbb{R}^3$). All hidden layers use *ReLU* activation, while the output layer employs different activation functions depending on the attribute. Table C.6 summarizes the details.

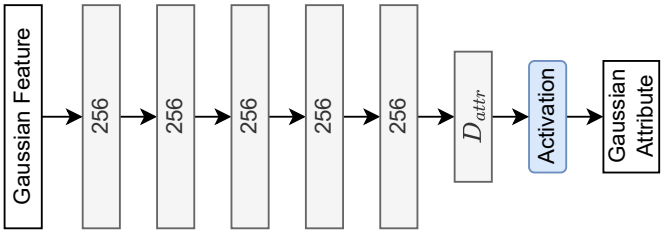


Figure C.8: Architecture of the decoder network.

Training

Training is performed in two stages. In stage one, the learning rates are set as follows: mapping network $\mathcal{D}$ (1e-5), multi-level latent field (5e-3), position offsets decoder (2e-4), scales decoder (1e-4), rotations decoder (1e-4), SH coefficients decoder (1e-4), and alphas decoder (1e-4). We train for 15,000 steps, with the learning rates reduced by half at step 10,000. In stage two, the learning rates are set as follows: multi-level latent field (5e-3), position offsets decoder (2e-4), scales

Table C.6: Summary of decoder networks. Each head predicts a Gaussian attribute, followed by its respective activation.

Network	D_{attr}	**Activation**
Position Offsets: $\Delta\mu$	3	scaled by 0.01
Rotations: $\mathbf{q}$	4	ℓ_2 normalization (quaternion)
Scales: $\mathbf{s}$	3	$\exp(\cdot) \times 0.0009$
Alphas: α	1	$\sigma(\cdot)$ (sigmoid)
SH Coefficients: $\mathbf{c}$	3	identity (no activation)

decoder (1e-4), rotations decoder (1e-4), SH coefficients decoder (1e-4), and alphas decoder (1e-4). We train for 30,000 steps, with the learning rates halved at steps 20,000 and 25,000.

C.2 ArchitectHead Additional Results

Per-Subject Self-Reenactment Results

We provide detailed evaluation results for each subject of the datasets in Tables C.7 and C.8.

Table C.7: Quantitative comparisons of self-reenactment results on the PointAvatar [239] dataset. Per-subject results are reported, with the best value of each evaluation metric in each column highlighted in bold.

Method	**Subject**	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓
GaussianAvatars (CVPR'24) [145]	1	0.064238	24.311394	0.888906	0.132315
	2	0.044746	27.840605	0.918122	0.137628
	3	0.035400	25.876420	0.940552	0.066730
FlashAvatar (CVPR'24) [210]	1	0.060337	25.321769	0.889419	0.086785
	2	0.039521	30.044333	0.917574	0.082006
	3	0.033305	26.302053	0.937210	0.041791
RGBAvatar (CVPR'25) [108]	1	0.070550	25.914843	0.884510	0.110350
	2	0.060199	27.439313	0.889355	0.138269
	3	**0.028802**	**30.964819**	**0.947895**	0.051413
Gaussian Dejavu (WACV'25) [219]	1	0.057501	26.603426	0.906823	0.075991
	2	0.047117	29.381665	0.917544	0.080533
	3	0.030174	27.471601	0.945235	0.038314
ArchitectHead (Ours)	1	**0.051053**	**27.158812**	**0.914353**	**0.068233**
	2	**0.035130**	**30.705131**	**0.923910**	**0.071576**
	3	0.028992	27.999216	0.946682	**0.034507**

Table C.8: Quantitative comparisons of self-reenactment results on INSTA [249] dataset. Per-subject results are reported, with the best value of each evaluation metric in each column highlighted in bold.

		Subject								
Method	**Metric**	bala	biden	justin	malte_1	nf_01	nf_03	obama	person_0004	wojtek_1
GaussianAvatars (CVPR'24) [145]	L1 ↓	0.029	0.024	0.023	0.028	0.036	0.041	0.030	0.034	0.022
	PSNR ↑	25.977	28.684	27.591	27.709	26.575	25.759	26.929	24.746	28.840
	SSIM ↑	0.937	0.956	0.962	0.941	0.940	0.922	0.948	0.936	0.959
	LPIPS ↓	0.080	0.050	0.051	0.062	0.088	0.095	0.052	0.097	0.049
FlashAvatar (CVPR'24) [210]	L1 ↓	0.028	0.022	0.023	0.025	0.036	0.038	0.024	0.026	0.020
	PSNR ↑	28.303	29.751	27.509	28.891	26.682	26.676	28.785	28.097	30.561
	SSIM ↑	0.927	0.962	0.963	0.946	0.938	0.922	0.958	0.943	0.960
	LPIPS ↓	0.035	0.027	0.033	0.029	0.064	0.058	0.029	0.050	0.023
RGBAvatar (CVPR'25) [108]	L1 ↓	0.058	0.020	0.033	0.040	0.056	0.032	0.027	0.046	0.024
	PSNR ↑	21.603	**31.945**	25.397	27.354	24.503	29.310	30.047	27.779	30.310
	SSIM ↑	0.869	0.957	0.943	0.903	0.895	0.939	0.948	0.924	0.941
	LPIPS ↓	0.077	0.034	0.049	0.061	0.095	0.065	0.038	0.080	0.037
Gaussian Dejavu (WACV'25) [219]	L1 ↓	**0.021**	**0.020**	**0.019**	0.025	0.033	0.033	0.022	0.026	0.016
	PSNR ↑	30.451	30.997	29.038	28.975	27.836	28.740	30.472	28.178	32.814
	SSIM ↑	**0.956**	0.968	**0.970**	0.949	0.945	0.938	0.963	0.945	0.973
	LPIPS ↓	0.024	0.024	**0.030**	0.028	0.060	0.053	0.026	0.049	0.017
ArchitectHead (Ours)	L1 ↓	0.022	**0.020**	0.020	**0.022**	**0.032**	**0.028**	**0.021**	**0.023**	**0.015**
	PSNR ↑	**30.986**	31.195	**29.521**	**30.537**	**28.399**	**29.380**	**30.894**	**29.305**	**33.287**
	SSIM ↑	0.955	**0.970**	0.968	**0.956**	**0.950**	**0.949**	**0.966**	**0.950**	**0.976**
	LPIPS ↓	**0.023**	**0.022**	0.036	**0.024**	**0.059**	**0.045**	**0.025**	**0.044**	**0.015**

More Cross-Reenactment Results

We provide more cross-reenactment results in Figure C.9.

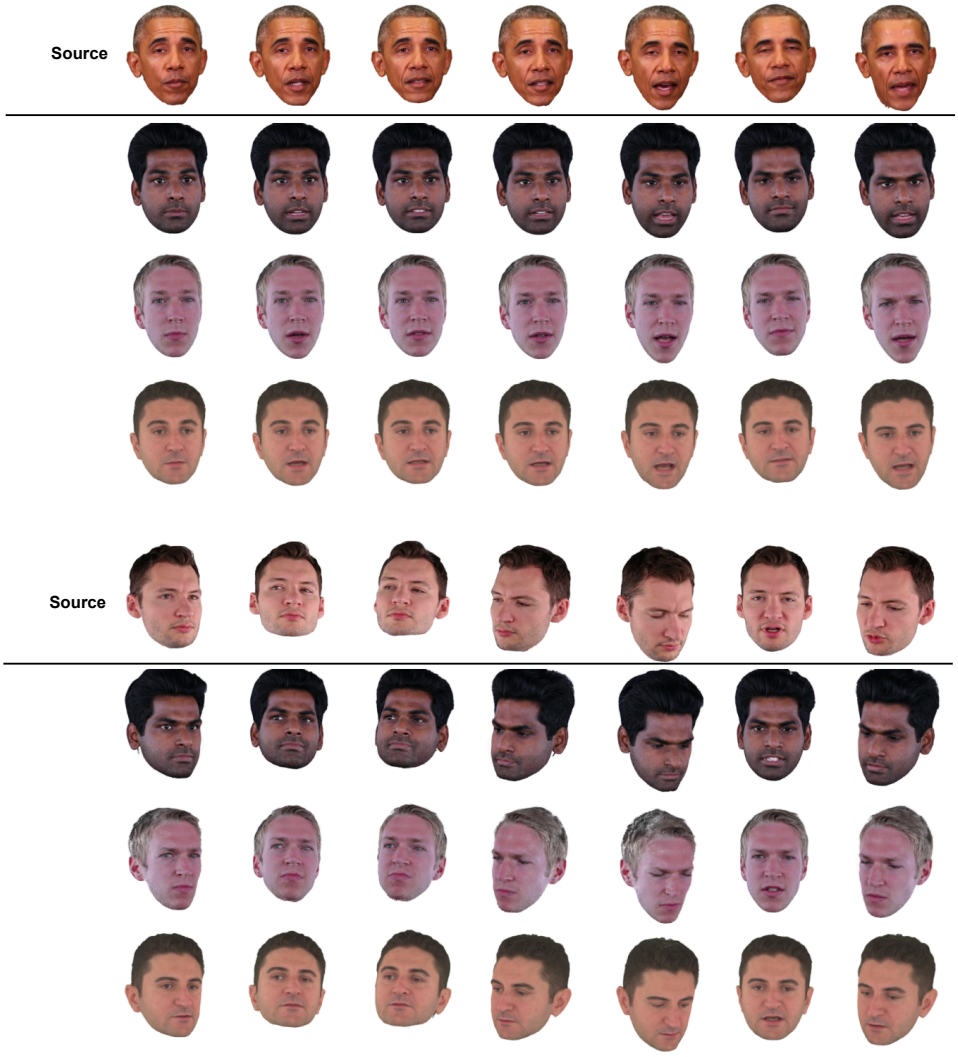


Figure C.9: Cross-identity reenactment results.

Varying Level of Details

We compare results across three LODs (0, 0.5, and 1.0) and different precisions (fp16 and fp32) during inference, while training is performed in full precision. The results are summarized in Tables C.9 and C.10. The differences are negligible for most metrics, with only slight variation in PSNR, indicating that using the trained

avatars in half precision does not noticeably degrade quality. Figure C.10 presents self-reenactment results at LODs 0 and 1.0. When zoomed in, some artifacts and loss of details are visible in the LOD 1.0 results. Overall, the quality at the lowest LOD (1.0) remains reasonable.

Table C.9: ArchitectHead on **PointAvatar** dataset, averaged over 3 subjects. Best values of different precisions under the same LOD level are highlighted.

Method (LOD)	**L1** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
ArchitectHead-fp16 (0.0)	0.038	28.615	0.928	0.058
ArchitectHead-fp16 (0.5)	0.039	28.508	0.927	0.061
ArchitectHead-fp16 (1.0)	0.041	28.337	0.922	0.072
ArchitectHead-fp32 (0.0)	0.038	28.621	0.928	0.058
ArchitectHead-fp32 (0.5)	0.039	28.514	0.927	0.061
ArchitectHead-fp32 (1.0)	0.041	28.342	0.922	0.072

Table C.10: ArchitectHead on **INSTA** dataset, averaged over 9 subjects. Best values of different precisions under the same LOD level are highlighted.

Method (LOD)	**L1** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
ArchitectHead-fp16 (0.0)	0.022	30.379	0.960	0.032
ArchitectHead-fp16 (0.5)	0.023	30.303	0.957	0.035
ArchitectHead-fp16 (1.0)	0.025	30.131	0.951	0.043
ArchitectHead-fp32 (0.0)	0.022	30.389	0.960	0.032
ArchitectHead-fp32 (0.5)	0.023	30.313	0.957	0.035
ArchitectHead-fp32 (1.0)	0.025	30.139	0.951	0.043

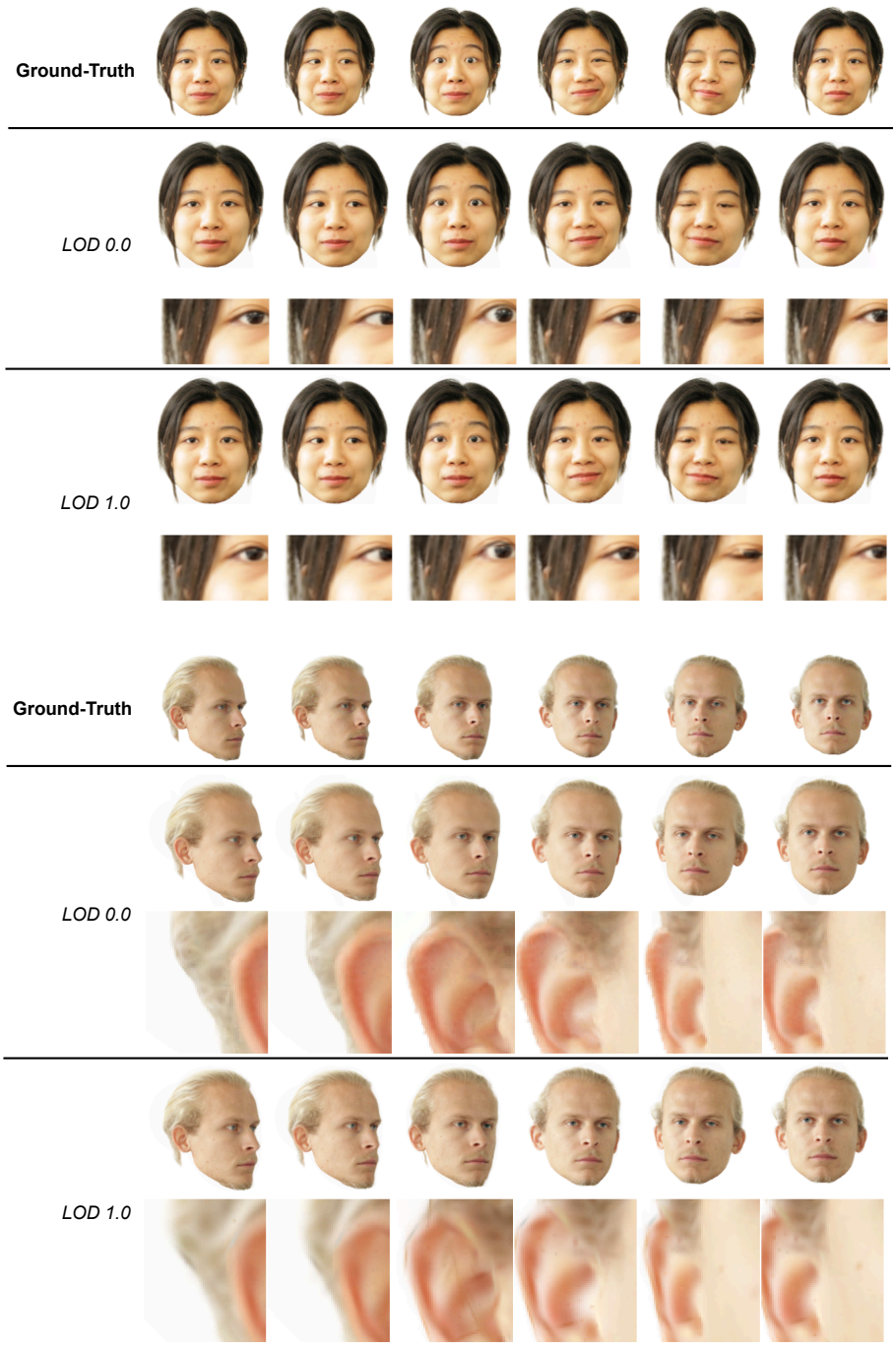


Figure C.10: Self-reenactment results with LOD 0 and LOD 1.0.