AI-Assisted Pipeline for 3D Face Avatar Generation

Amin Fadaeinejad

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

September 12, 2024

Abstract

Filling virtual environments with realistic-looking avatars is essential for games, film production, and virtual reality. Creating a fun and engaging experience requires a wide variety of different-looking avatars. There are two main methods to create realistic-looking avatars. One is to scan a real person's face using a light room. The second is for the artist/designer to create the avatar manually using advanced tools. Both of these approaches are expensive in terms of time, computing, and human labour. In this thesis, we leveraged the power of generative models to automate the process of creating face avatars. Generative models such as Generative Adversarial Networks (GAN) and Variational Auto-Encoders (VAE) are used throughout our pipeline. In the process of creating face avatars, we can control the appearance of the face avatar on three different levels: Level 1: Enabling control over the face shape geometry. Level 2: Controlling the skill colour of the avatar. Level 3: Editing fine-grained details of the avatar, such as adding/removing beard, wrinkles, and pores. These three steps provide enough control to the artist to create a face avatar suited for the case. This pipeline can be used with other existing tools (such as MOSAR [10]), enabling the control of face avatars extracted from a 2D image.

Acknowledgements

I am profoundly grateful to my advisors, Niko Troje and Marcus Brubaker, for their unwavering support and invaluable guidance throughout my Master's journey. Their mentorship not only deepened my understanding of research beyond the pursuit of results but also instilled in me the importance of rigorous inquiry and curiosity. Their enthusiasm and confidence in my abilities have been a constant source of motivation, reinforcing my belief that I am on the right path. I am truly thankful for their mentorship and its significant impact on my academic and personal growth.

I would also like to express my sincere gratitude to Ubisoft for the unique opportunity to integrate my academic research with practical, real-world applications. This collaboration has been a pivotal and enriching part of my Master's program. Special thanks to my manager, Abdallah Dib, whose guidance, support, and trust in my work have been invaluable. His insights, encouragement, and leadership have significantly contributed to my professional development and the success of my project.

My colleagues at Ubisoft deserve a heartfelt thank you for their support, collaboration, and the inspiring environment they provided. Working alongside such a talented and creative team has greatly enhanced my experience and contributed to my professional and personal growth.

Thank you to everyone who has played a part in this remarkable journey. Your contributions, support, and faith in my potential have been invaluable, and I eagerly anticipate carrying forward the lessons and memories we have shared.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Video games are one of the many entertainment sources available today. To create a fun and engaging experience for users, it is essential to develop an environment with realistic-looking avatars. It is also important for games to have a wide variety of diverse-looking avatars. Video games or virtual reality environments with a single-looking avatar are not appealing. However, the creation of diverse and realistic-looking face avatars on a large scale presents significant challenges. This process requires a considerable amount of time and expertise in facial modelling. These challenges drive the search for more efficient methods to produce and modify 3D head assets.

A common method for generating head assets uses 3D Morphable Models [4], which are statistical models for producing diverse 3D facial geometries. These techniques develop a statistical model from a collection of head scans [12]. New head representations can be generated by blending the base elements from this model. However, the face inference it produces often lacks detailed features. Researchers have explored approaches

using generative models such as Generative Adversarial Networks (GANs) [19] to create more detailed face avatars [35, 49]. These approaches aim to improve the details of head textures by using high-quality texture maps, such as colour, specular, and normal maps.

However, experience has shown that GANs are generally difficult to train (challenges such as mode collapse, non-convergence, and instability problems) and are prone to produce artifacts during generation. One way to generate 3D assets while leveraging Convolutional Neural Networks (CNN) is to represent geometry in a canonical space by projecting the 3D vertex positions onto a 2D UV map, which has its advantages and drawbacks. As an advantage, it is much easier for a GAN to generate 2D data rather than 3D due to the complexity of the structure of 3D data. One main drawback would be that it is not possible to unwrap a 3D head geometry to 2D without any cuts. This causes some vertices that are close in 3D space to appear far in the 2D UV space. Additionally, depending on the UV parametrization, vertices of facial features, such as the mouth and eye contours, may be close in UV space but far in 3D space, which results in artifacts during generation. For this reason, most methods that estimate geometry in the UV space rely on post-processing steps. For instance, in [35], a separate linear model is applied to the problematic areas (eyes and mouth), which are later integrated into the geometry produced by the non-linear generator. In contrast, methods that use Graph Neural Networks (GNNs) [50] work directly on the vertex-level geometry, avoiding the limitations associated with GANs. GNN-based generative methods can produce complete head geometry without the need for additional post-processing because they operate at the vertex level. However, these methods are not ideal for generating detailed texture maps, which are typically represented as high-resolution images (e.g., 4K or 8K in modern games and films).

Recent approaches in research papers do not allow users to interact with the face instance for precise manipulation of the produced head. For instance, methods such as [34, 33, 46] are conditioned on images, which requires manipulating face attributes in the input image space in the hope that the network will accurately replicate the details. Recent systems conditioned on text prompts [67, 62] allow users to change face attributes on a global level but are not capable of producing details and the exact location (e.g., you can't precisely tell the system where you want a detail to exist based on just a prompt). Moreover, sometimes words can fail to precisely describe a skin tone or facial features, such as the shape of the eyebrows, leading to extensive trial and error. This limitation restricts the utility of these techniques in high-end tools for projects with particular artistic requirements and constraints. However, there are some paid software programs that provide a certain level of control for users to interact with the face instance, such as MetaHuman, iClone, and others.

In this work, we introduce a novel framework for generating 3D face assets tailored to provide artists with precise control over the generated assets. Our approach combines a GNN-based model for generating geometry at the vertex level coupled with a texture generator that uses a CNN-based network to generate intrinsic 2D texture maps.

Notably, we propose a geometry-aware texture generation process that is specifically designed to learn the correlations between head geometry and skin texture. In addition to the ability to generate different geometry, our work provides artists with more control over the final output. Our texture generation pipeline includes the following capabilities:

1. Precise skin tone control, which controls the skin colour of the generated face avatar.

2. Add/remove fine-grained details such as beard, wrinkles, and pores.

This is achieved using a novel method that separates the high-frequency and low-frequency attributes of the skin, allowing for separate control over both skin tone and fine-grained details. Additionally, these modifications are cohesively propagated across the intrinsic texture maps (colour, specular, and normal). This approach significantly saves artists time, as they only need to edit a single map to affect all related aspects of the texture. Figure 3.2 displays a high-level view of the proposed pipeline.

In this thesis, we make the following contributions:

- We developed a GNN-based Variational Autoencoder (VAE) capable of various face geometry without requiring any post-processing steps.
- We combine a geometry-aware texture generator with the GNN-based VAE(the one responsible for generating the face geometry) to generate the essential texture maps (colour, specular, and normal). This enables the creation of high-quality head assets that match the identity of the face geometry.
- We present a data-driven method that allows precise skin tone manipulation while trying to preserve other texture aspects.
- We introduce a novel method that simplifies the editing of fine-grained details on intrinsic texture maps. This approach allows artists to modify a single texture map using standard image editing tools, which automatically propagates those modifications across all intrinsic texture maps. This streamlined process enhances efficiency and consistency in texture editing.

Chapter 2

Background

In this section, we will review some essential concepts and background knowledge necessary to understand this thesis. Initially, we will discuss various face representations, the methods used to capture them, and the dataset that was captured using that representation. Subsequently, we will explain some common deep generative models frequently utilized in this thesis and explain their functionality and training scheme. Finally, we will cover previous works that shared a goal similar to ours, which is developing a pipeline capable of generating face avatars.

There are many different 3D face representations; however, in this section, we will discuss the three common ones. We will also mention datasets previously introduced in this field, detailing their construction and providing additional information about them.

- **Basel Face Model**: The Basel Face Model (BFM) [48] is a statistical 3D face model that represents human facial shapes and textures. It is created using 3D scans of human faces and is often used in computer vision, computer graphics, and ma-

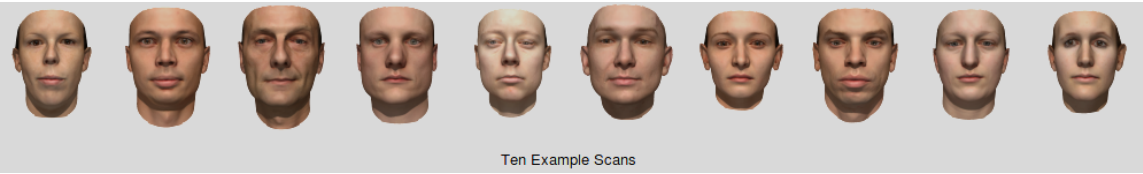


Figure 2.1: Samples of Basel Face model

chine learning applications. The same group also captured a large dataset of 3D facial scans, capturing the variations in human facial geometry and appearance. This dataset includes a collection of vertices that define the 3D shape of the face, as well as texture information that represents the surface appearance. A fundamental way to create new face instances using the Basel Face Model is to use Principal Component Analysis (PCA) [14] to model the statistical variations in facial shapes and textures. PCA is a method that identifies the most significant variations within a dataset and represents them as a set of orthogonal axes known as principal components. Figure 2.1 illustrates some examples of this face model.

- **FaceWarehouse**: FaceWarehouse [5] is a notable initiative in the realm of facial modelling and computer vision. It is also a 3D facial expression database that provides the facial geometry of 150 subjects aged 7-80, covering a wide range of ethnic backgrounds. The authors captured the RGBD data of different expressions, including the neutral expression and 19 other expressions such as mouth-opening, smile, kiss, etc. For each RGBD scan, a set of facial feature points on the colour image, such as eye corners, mouth contour, and nose tip, are automatically localized and manually adjusted if better accuracy is required. Its primary objective is to capture statistical variations in facial shape and texture through an analysis of a 3D

facial scan dataset. FaceWarehouse is primarily a diverse dataset characterized by high-resolution 3D facial meshes that precisely define the geometric aspects of facial structure. In addition, this dataset also consists of corresponding texture feature map information for understanding both the physical structure and visual appearance of a human face. One of the key advantages of FaceWarehouse is its use of advanced statistical methods, such as Principal Component Analysis (PCA), to identify and represent the complex variations in facial characteristics. By analyzing this dataset, FaceWarehouse employs methods such as PCA to extract patterns and variations in facial shape and texture. These components are foundational for generating a wide spectrum of facial shapes and appearances. Compared with the Basel database, every person in the FaceWarehouse database has a much richer matching collection of expressions, enabling the depiction of most human facial actions. This versatility makes FaceWarehouse an invaluable resource for various applications, including facial image manipulation, face component transfer, real-time performance-based facial image animation, and facial animation. Figure 2.2 showcases some face model samples from the FaceWarehouse dataset.

- **Faces Learned with an Articulated Model and Expressions (FLAME)**: Faces Learned with an Articulated Model and Expressions (FLAME) [36] is an advanced 3D head representation that offers a practical balance between high-end and basic facial modelling techniques. FLAME achieves this by leveraging a large dataset of 3D scans, which are accurately aligned, and using a combination of linear shape space with expressive blend shapes. The face's linear shape representation is trained from 3,800 scans of human heads. In addition to the shape, FLAME also learns pose

Figure 2.2: Samples of FaceWarehouse dataset. Top: The input scan, middle: Input depth map, bottom: Refined face mesh

and expression-dependent articulation from a 4D face sequence in the D3DFACS dataset alongside the 4D sequence. Over 33,000 scans were required to learn the pose and expression-dependent aspects. Overall, FLAME is low-dimensional yet more expressive than both the FaceWarehouse and Basel Face Model. This is evident when comparing FLAME to these face representations by fitting them to static 3D scans and 4D sequences using the same optimization method. FLAME demonstrates significantly higher accuracy compared to the other face representations mentioned, and it is also available for research purposes. FaceWarehouse is the only publicly available 3D face database that offers a large number of facial expressions and comes with template meshes aligned to raw scan data (from a depth sensor). The D3DFACS dataset, while having much higher-quality scans, does not contain aligned meshes. Registering such 4D data presents an additional challenge. Figure 2.3 shows some

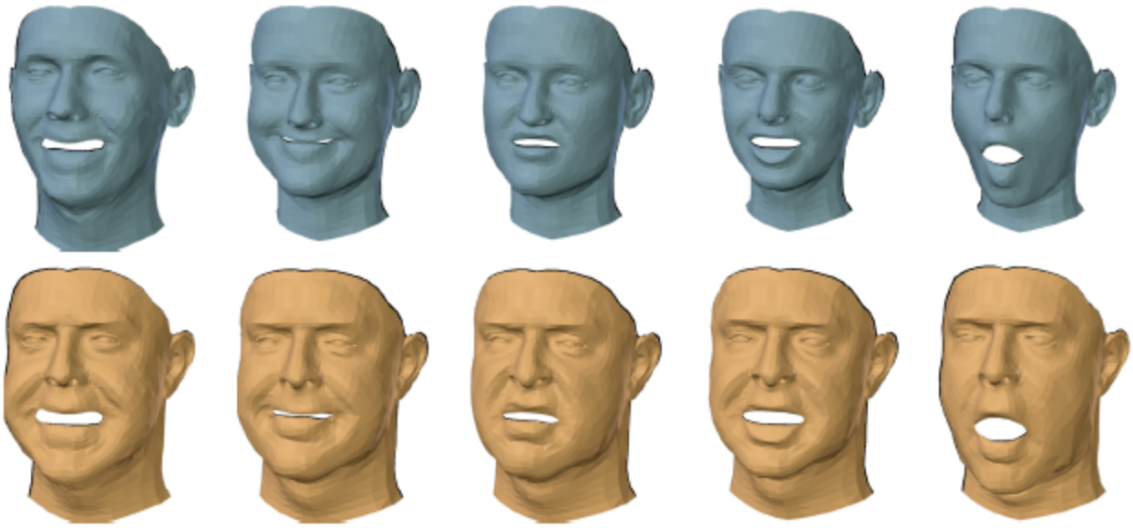


Figure 2.3: Samples of FLAME Face model

face model samples from the FLAME.

2.1 Generative models

Generative models have shown impressive results in generative tasks, such as image generation. Designed to produce new data samples that resemble a given dataset, they have gained popularity in the domain of computer vision and graphics due to their remarkable ability to generate realistic images and videos. Among the various types of generative models, Variational Autoencoders (VAEs) [32], Generative Adversarial Networks (GANs) [19], and Diffusion models [22] stand out as leading technologies in this field. In this thesis, we primarily used Variational Autoencoders (VAE) and Generative Adversarial Networks (GANs), which we will explore in further detail in subsequent sections.

2.1.1 Variational Auto-Encoder (VAE)

Variational Autoencoders (VAEs) are a class of generative models that are based on the principles of Bayesian inference. They aim to learn the underlying probability distribution of a given dataset, allowing them to generate new data points that resemble the original dataset. VAEs are particularly useful in unsupervised learning tasks, where the goal is to understand the structure of the input data without any explicit labels. At a high level, a VAE consists of two main components: the encoder and the decoder.

- **Encoder:** The encoder maps the input data x to a latent representation z, capturing the essential features of the data in a compressed form. The encoder is a neural network. The size of the latent representation must be smaller than the input size.
- **Decoder:** The decoder then reconstructs the input data (denoted as $\hat{x}$) from the latent representation z, attempting to generate an output that is as close as possible to the original input.

Mathematical Formulation

The core idea of a VAE is to model the latent variable z using a probability distribution $q(z|x)$, which approximates the true posterior distribution $p(z|x)$. The encoder parameterizes $q(z|x)$ as a Gaussian distribution with mean $\mu(x)$ and variance $\sigma^2(x)$, both of which are functions of the input data generally represented by a neural network. The VAE optimizes the variational lower bound—or evidence lower bound (ELBO)—on the marginal likelihood of the observed data. The ELBO can be expressed as:

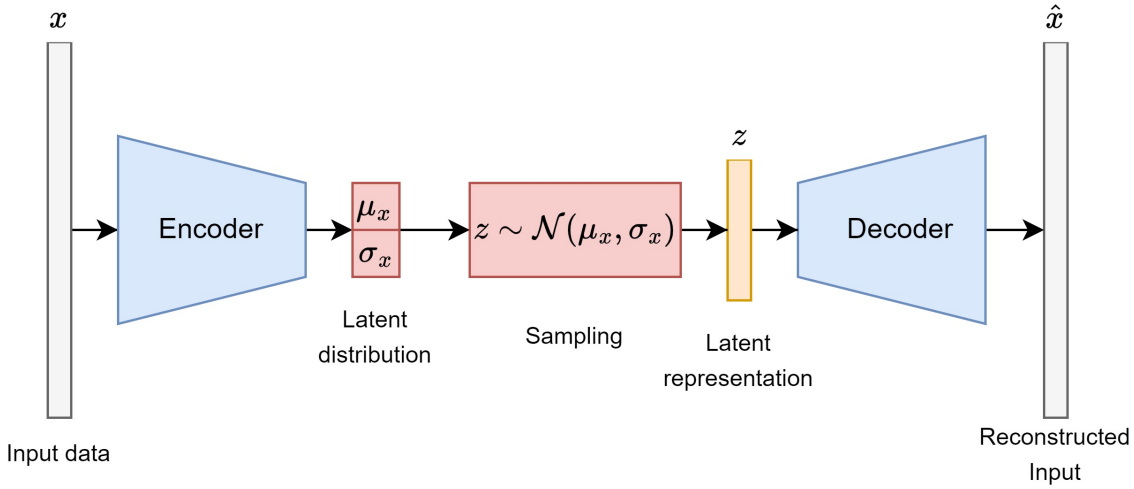


Figure 2.4: VAE architecture

$$\text{ELBO} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}[q(z|x)||p(z)] \tag{2.1}$$

where:

- $\mathbb{E}_{q(z|x)}[\log p(x|z)]$ is the expected log-likelihood of the data given the latent variables, which encourages the decoder to accurately reconstruct the input data from the latent variables.
- $D_{KL}[q(z|x)||p(z)]$ is the Kullback-Leibler (KL) divergence between the approximate posterior $q(z|x)$ and the prior distribution $p(z)$, which acts as a regularization term, encouraging the approximate posterior to be close to the prior.

The first term of the ELBO ensures that the reconstructed data is similar to the original data, promoting the accuracy of the reconstruction. The second term regularizes the

learned latent space by penalizing deviations of the learned distribution $q(z|x)$ from the prior distribution $p(z)$, typically assumed to be a standard Gaussian distribution. In practice, to optimize the ELBO, a reparameterization trick is used to allow gradients to flow through the stochastic nodes of the network. Instead of directly sampling z from $q(z|x)$, z is expressed as a deterministic function of x and some random noise ϵ, as follows:

$$z = \mu(x) + \sigma(x) \bullet \epsilon \tag{2.2}$$

where $\epsilon \sim \mathcal{N}(0, I)$ and $\bullet$ denotes element-wise multiplication. This formulation allows for efficient gradient-based optimization of the ELBO with respect to the parameters of the encoder and decoder networks, facilitating the training of VAEs through backpropagation. Figure 2.4 illustrates the architecture of the VAE, where the parameters of the latent distribution (μ_x, σ_x) are derived by the encoder network. Subsequently, the latent representation z is sampled from a Gaussian distribution with parameters (μ_x, σ_x). Finally, the decoder model reconstructs the input by processing the latent representation z.

2.1.2 Generative Adversarial Network (GAN)

Generative Adversarial Networks (GANs) are a type of generative model used to produce new data samples that resemble a given dataset. They consist of two neural networks, the generator and the discriminator, which are trained simultaneously in an adversarial fashion. Here's a detailed explanation of each component and how they work together:

- **Generator**: The generator is a deep neural network that takes in a random noise vector as input and generates synthetic data that resembles the training data. Its

purpose is to create data that is indistinguishable from real data.

- **Discriminator**: The discriminator, on the other hand, is like a binary classifier that evaluates whether a given input is real (from the actual dataset) or fake (generated by the generator). Its goal is to correctly distinguish between real and synthetic data.

Training Process: GANs have a unique way of training. During training, the generator and the discriminator are trained simultaneously through adversarial training. The generator tries to generate more realistic images to fool the discriminator, and the discriminator tries to learn the ability to classify real and fake data samples. Ideally, this training process reaches an equilibrium where the generator produces data that is indistinguishable from real data, and the discriminator cannot reliably differentiate between real and generated samples.

Loss: The GANs loss function can be viewed in equation 2.3, where the R is the real image distribution and $\mathcal{N}$ is the Normal distribution. The standard GAN loss function, also known as min-max loss, was first described in Goodfellow *et al*. [19]. The generator tries to minimize this function while the discriminator tries to maximize it. Looking at it as a min-max game, this formulation of the loss seemed effective. In practice, it saturates for the generator, meaning that the generator quite frequently stops training if it doesn't catch up with the discriminator.

$$\mathcal{L} = \mathbb{E}_{x\sim R}[\log(D(x))] + \mathbb{E}_{z\sim\mathcal{N}}[\log(1 - D(G(z)))] \tag{2.3}$$

The Standard GAN loss function can further be categorized into two parts: Discrimi-

nator loss and Generator loss.

- **Discriminator loss**: While the discriminator is trained, it classifies both the real data and the fake data from the generator. It penalizes itself for misclassifying a real instance as fake, or a fake instance (created by the generator) as real, by maximizing the below function. In equation 2.4 $\log(D(x))$ refers to the probability that the generator is rightly classifying the real image, maximizing $\log(1 - D(G(z)))$ would help it to correctly label the fake image that comes from the generator.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))] \tag{2.4}$$

- **Generator loss**: While the generator is trained, it samples random noise and produces an output from that noise. The output then goes through the discriminator and gets classified as either "Real" or "Fake" based on the ability of the discriminator to tell one from the other. The generator loss is then calculated from the discriminator's classification – it gets rewarded if it successfully fools the discriminator and gets penalized otherwise.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)}))) \tag{2.5}$$

Different GANs: In this section, we will explain the different existing GANs.

- **StyleGAN**: StyleGAN [28, 30, 29, 31] is one of the members of the GAN family that uses a specific stylization technique for generating realistic-looking images. Figure 2.6 shows 3 of the most well-known models from the StyleGAN family, with

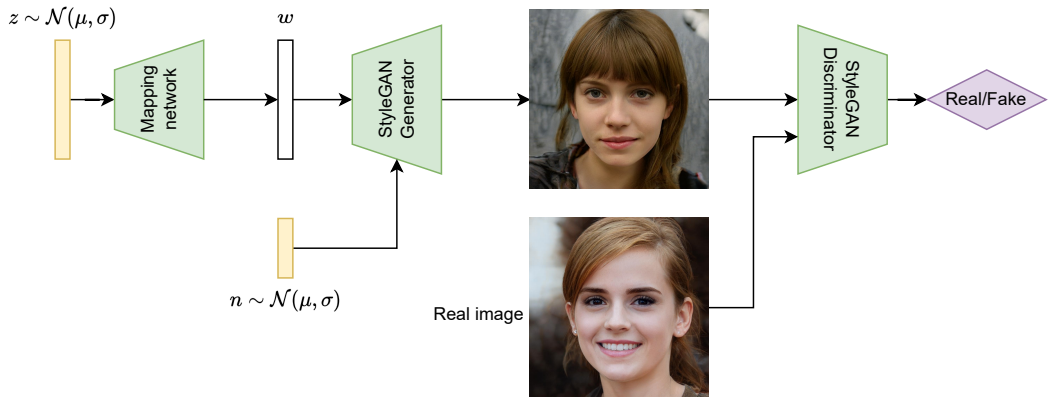


Figure 2.5: High-level overview of StyleGAN

their architectural differences. Primarily, StyleGAN was designed to create realistic images while manipulating and controlling certain features or styles of the image. These styles associated with the generated images could be features like colour, texture, pose, etc. Figure 2.5 shows an overview of the training scheme of the StyleGAN. Traditionally, a GAN generator takes a noise vector z as an input, and then the model returns the output (it could be an image or any sort of data). In StyleGAN, instead of feeding the noise vector z directly into the generator, the noise vector z goes through a mapping network, producing an intermediate style representation w, which has information regarding the style of the content. Subsequently, w gets injected through an operation called adaptive instance normalization (AdaIN) [23] into multiple layers of the Synthesis network of the StyleGAN which will then result in the generation of the fake image. Additionally, in order to add more randomness and better generalization, an extra random noise is passed to the Synthesis network. To get a better perception of how StyleGAN works, check Figure 2.5.

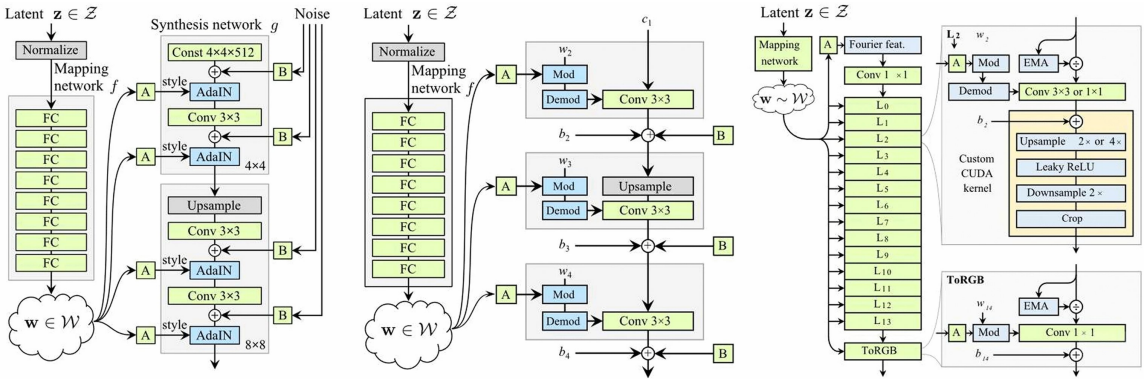


Figure 2.6: Architectures of StyleGAN generators from left to right: (a) StyleGAN [28], (b) StyleGAN2 [30], (c) StyleGAN3 [31]

- **Pix2Pix** Pix2Pix [25], also known as the image-to-image translation network, is a type of conditional generative adversarial network (cGAN) specifically designed for image-to-image translation tasks. This model architecture enables the conversion of an input image into a corresponding output image, following a certain learned transformation pattern. It's widely used for tasks such as photo enhancement, colourization, style transfer, and more. Here's a breakdown of how Pix2Pix operates:

  - **Generator:** The generator takes an image (from domain A) as input and generates a corresponding image that attempts to resemble a target style or format (domain B). It typically uses a U-Net architecture, which is effective for these tasks because it allows for the preservation of spatial hierarchies between input and output through skip connections.

  - **Discriminator:** The discriminator's job is to differentiate between the real images (drawn from the training dataset) and the fake images (generated by the

generator). It acts like a critic that helps the generator produce better outputs. The discriminator usually has a PatchGAN [9] structure, which means it evaluates the real/fake status of different parts (patches) of the image separately rather than classifying the entire image at once.

The training process for training this Pix2Pix model is as follows:

- **Adversarial Training:** In the adversarial training process, the generator and discriminator are trained simultaneously. The generator tries to produce images that are indistinguishable from the real images to fool the discriminator. The discriminator, in turn, learns to become better at distinguishing the fake images from the real ones.
- **Objective Function:** The loss function used in Pix2Pix is a combination of a traditional GAN loss and an L1 loss (which encourages the generated images to be close to the target images in terms of pixel-wise content). The L1 loss helps in reducing blurring and preserving structural details in the generated images.

Figure 2.7 shows the type of output the Pix2Pix model generates, given a specific input. Additionally, the ground truth data is also displayed for comparison.

Architecture: Figure 2.8 shows the architecture that has been used for pix2pix is displayed. The generator is inspired by U-net [51], which is well-known for its performance for image-to-image tasks. For the discriminator, unlike traditional discriminators, which attempt to classify an entire image as real or fake, the PatchGAN classifies patches of an image. This method allows the discriminator to focus on

Figure 2.7: left: Input Image, middle: Ground Truth, right: Predicted Image

high-frequency details, effectively teaching the generator to pay attention to fine details and textures, thereby producing more realistic images.

2.2 Metrics

In this section, we will briefly explain some well-known metrics that are used to evaluate the fidelity and diversity of generated assets. In the end, we will explain which ones are useful for our case and why. In the section 4, we will show the evaluation of the metrics and how we used them.

2.2.1 Fréchet inception distance (FID)

The most well-known metric to evaluate generated images is Fréchet inception distance (FID)[21]. FID is a metric for quantifying the fidelity and diversity of images generated by generative models. Fidelity in the sense that generated images would look like real images of people. Diversity in the sense that the generated images cover a large distribution and

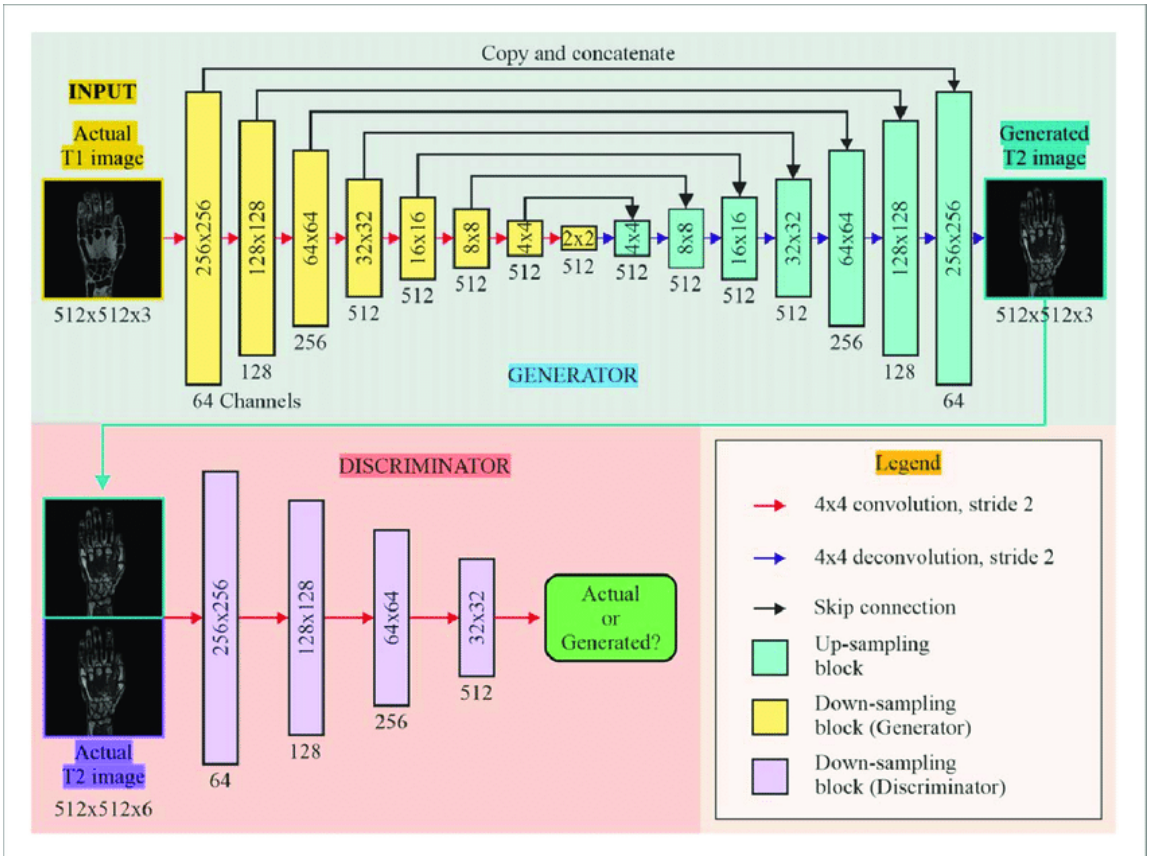


Figure 2.8: Diagram illustrating the Pix2Pix network architecture.

produce the same results repeatedly. FID is generally used for analyzing images and not text, sounds, or other modalities. Other related metrics are being developed for these domains. FID assesses visual quality and diversity well within a single metric. A lower score can be measured when generated images are more like real images. Equation 2.6 shows how the FID score is calculated, where the Inception-v3 [53] pre-trained model is used to extract the feature vector of real images and generated images. μ_r and μ_g are the mean feature vectors of the real and generated images. Notably, Σ_r and Σ_g are the

covariance matrix of the feature vectors of the real and generated samples.

$$\text{FID} = ||\mu_r - \mu_g||^2 + \text{tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \tag{2.6}$$

2.2.2 Kernel Inception Distance (KID)

One other similar metric is Kernel Inception Distance (KID) [21]. KID is defined as the squared Maximum Mean Discrepancy (MMD) distance between Inception representations, employing a polynomial kernel, $k(x, y) = (\frac{1}{d}x^T y + 1)^3$ where d is the representation dimension. Similar to the Frechet Inception Distance (FID), KID utilizes a pre-trained Inception-v3[53] network. However, unlike FID, KID does not assume a parametric form for the distribution of activations and is unbiased. The lower the KID score, the better the performance of the generative model (similar to FID).

2.2.3 Inception Score (IS)

Inception score (IS)[53] is one other well-known metric to evaluate how realistic an image is. Unlike the earlier FID and KID, which compare the distribution of generated images with the distribution of a set of real images, IS evaluates only the distribution of generated images. It applies the Inception model to every generated image to get the conditional label distribution $p(y|x)$ based on two assumptions: (1) Images that contain meaningful objects should have conditional label distribution $p(y|x)$ with low entropy, i.e., photos having a high probability belonging to one class. (2) The model generates varied images, so the marginal $\int p(y|x = G(z))dz$ should have high entropy, i.e., a good generative

model should output different classes uniformly. Equation 2.7 shows the formula of this score where:

- $x \sim p_G$ are the generated samples by the generator.
- $p(y|x)$ is the conditional label distribution of generated sample x.
- $p(y)$ is the average conditional label distribution of all generated assets.
- $D_{KL}(P||Q)$ is the Kullback–Leibler divergence, also called relative entropy, a measure of how one probability distribution is different from a second, reference probability distribution (follow equation 2.8 for more details).

However, one of the disadvantages of this metric is that since Inception-V3 was trained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [52] dataset, it performs best with samples that belong to a class within that dataset. This implies that if an image does not belong to any class in ImageNet, the Inception Score (IS) may not be a representative metric.

$$IS = e^{\mathbb{E}_{x \sim p_G} D_{KL}(p(y|x)||p(y)} \tag{2.7}$$

$$D_{KL}(P \parallel Q) = \sum_{i=1}^{n} P(i) \ln(\frac{P(i)}{Q(i)}) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} \mathrm{d}x \tag{2.8}$$

2.2.4 Metric Selection

In this thesis, we will use the FID to evaluate the quality of the generated assets. As mentioned in 2.2.3, the Inception Score (IS) is most effective when the generated image exists in the dataset. Since our dataset contains no human images, IS is not the best metric for our purposes. Additionally, our experiments with IS showed that its values do not correlate with image quality. For instance, IS was high when the model generated pure noise in the first epoch. This shows that IS is not a suitable metric for our evaluation.

2.3 Previous Approaches to Face Generation

As explained in section 2, the generation of face avatars has mostly relied on statistical approaches [4, 36, 16, 6] in the past. The process of generating new face avatars starts by calculating the Principal Component Analysis (PCA) on a collection of head scans. The resulting PCA basis can be used to sample new heads by changing the PCA coefficients for reconstructing a face. Due to the lack of clear interpretation on this basis, controlling or editing the generation process is not intuitive. Additionally, these generated heads tend to lack finer details, as high-frequency components aren't well captured by PCA. Deep Generative models [35, 49, 15] have been shown to produce more detailed heads compared to older PCA-based approaches [15]. These Deep Generative models can be split into two main categories. The first one leverages Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN) [19] operating on UV space [35]. The second uses Graph Neural networks (GNN) operating directly on vertex coordinates [2]. Similar to [35], the approaches using CNNs consider the generation of geometry and colour within

the 2D UV space; this, on the other hand, requires flattening the 3D geometry into UV space. However, flattening the geometry onto a 2D space without making cuts isn't feasible. As a result, some areas that are close together in 3D end up being far in the 2D representation. Conversely, some positions that are close on the 2D UV map can be far apart in the 3D mesh, such as the eyes and mouth. This leads to visible artifacts in the geometry; post-processing is generally applied to fix these artifacts as in [35]. The second approach for generative head models uses Graph Neural network (GNN) and auto-encoders to generate a complete head without requiring additional post-processing [50, 2]. However, as they operate on a vertex-level basis, the quality of the colour texture maps seems inferior to that of the colour maps generated from other CNN-based generative models.

Controllable generative models Controlling deep generative models has achieved remarkable advances for 2D portrait image editing, mainly because of the availability of a large set of 2D portrait image datasets such as [38, 28]. Such methods [1, 68, 44, 66, 65, 54, 55, 24, 37] allow for semantic editing of the face attributes on 2D portrait images. This is achieved by projecting the image onto the latent space of StyleGAN [30] and finding orthogonal directions in that space that allow for controlling various face attributes (such as hair, age, expression, and eyeglasses). Recently, works such as [59] used a pre-trained StyleGAN [31] to control the skin tone of a 2D portrait image. Although numerous works have tackled the problem of controlling generative models for 2D portrait images, there is a scarcity of work addressing this problem for 3D facial asset generation. When referring to 3D facial assets, we consider both the geometry (coordinates of vertices in 3D space) and the associated texture maps (including colour (albedo), specular, and normal maps) used for realistic rendering. Recent work such as StyleRig [58] drives a pre-trained GAN

network with 3D morphable model (3DMM) [4]. Following the same direction, Albedo-GAN [49] uses a pre-trained StyleGAN to generate a 2D image and then recover back the 3DMM parameters. These models allow controlling the expression, head pose, identity, and scene lighting of the generated image. However, they inherit the limitations of 3DMMs and do not provide artists with precise control. For instance, in the mentioned work, artists cannot manipulate and edit the texture maps and geometry directly to achieve their desired output. However, some other methods allow for limited control, generating a subject with a specific gender and age [35] or ethnicity and body mass [15]. In [43, 42], the artists control the texture generation model using a segmented feature map to specify colours and specify global attributes through tags. More recently, diffusion-based models [67, 62] proposed to produce realistic head avatars from text prompts. However, these methods lack fine-grained detail control over the generation process, making them less usable to artists.

In contrast to previous work, our approach offers fine-grained artistic control over a generative model designed to fit the artists' needs. This is achieved by designing a pipeline that takes detailed artist control at intermediate points of the generation process, enabling them to interact with generated assets by independently adjusting features to control the skin tone and fine-grained geometric details.

Skin tone colour control Precise adjustment of the skin colour of virtual characters is crucial for artists when designing a character from a specific racial group. Furthermore, tweaking skin tone can prove valuable in addressing biases towards underrepresented ethnicities. Moreover, this capability enables the creation of diversity and enhances the realism of the user experience. However, manipulating skin tone on a large scale can be

challenging and time-consuming.

The skin colour of human skin is determined by the levels of melanin and hemoglobin concentration in the epidermis layer [3, 20]. This information serves as a basis for modifying the skin tone of virtual characters. However, accurately capturing melanin and hemoglobin concentrations is a resource-intensive task that requires specialized equipment and procedures [17, 45]). Some work aims to estimate these properties from images. Pioneer works such as [61, 60] use independent component analysis (ICA) to estimate melanin and hemoglobin distributions in 2D portrait face images. In [11] proposed a parametric skin reflectance model based on melanin and hemoglobin concentrations, enabling control over skin colour. Nevertheless, understanding the complex relationship between melanin/hemoglobin concentrations and skin colour is challenging, with limited literature available on obtaining an explicit relationship between skin colour and the melanin-hemoglobin representation.

In this thesis, we propose:

- A Geometry generator capable of producing 3D face meshes using GNN-based models, which allows approximate control over the subject's physical attributes.
- A data-driven approach that enables control over the skin colour of a face avatar.
- A method that provides precise control over the intrinsic texture maps, giving artists and designers complete authority to modify (adding or removing fine details) the appearance of the face model.

The goal of the thesis is a pipeline for generating 3D face assets that can be edited and used for animation-based applications like video games.

Chapter 3

Method

In the methods section, we will review the dataset that was used and explain each of its features. Subsequently, we will describe some of our evaluation metrics, how they function, and how we employed them to assess our pipeline. Lastly, we will discuss the pipeline's architecture, detailing each component and its training process.

3.1 Dataset

For simplicity, we will use the same notation throughout this thesis. For this thesis, we used a dataset that was previously captured by the Ubisoft team, and I have no involvement in capturing it. This dataset consisted of 892 head scans captured using a light stage. The light stage setup involved a set of high-resolution cameras that surrounded the subjects' heads, ensuring complete image capturing from multiple angles. This setup precisely captured the subjects' facial features under various lighting conditions. The light stage generated detailed reflectance fields for each face by altering the lighting and recording

the imaging responses. These feature assets show how light interacts with different facial textures and geometries. Ultimately, this light stage will provide high-resolution texture maps and accurate 3D models of the heads. The detailed capture process enabled the collection of high-resolution data on facial features, which is essential for the realistic rendering and animation of the digital heads. Figure 3.1 showcases a single sample from this dataset, illustrating the level of detail and accuracy achieved through the light stage technology. For each subject in the dataset, we have the face geometry (vertices positions V), the colour map $\mathcal{C}$ at 4K resolution, the specular $\mathcal{S}$ and normal map $\mathcal{N}$. To obtain the smooth colour map $\mathcal{A}$, we apply a Principal Component Analysis (PCA) over the reflectance map $\mathcal{C}$ for the entire training dataset, retaining only the first 15 eigenvectors. Subsequently, we project each $\mathcal{C}$ onto the PCA basis. The result is a low-frequency image with the base skin colour of each subject. We obtain $\mathcal{H}$ using the Sobel operator [27], which captures high-frequency details such as folds, wrinkles, moles, and pores. We use a curated dataset of maps $\mathcal{M}$ for melanin-hemoglobin representation, based on results from a commercial tool[1].

In the following section, we will briefly explain each of the data assets (mesh geometry and texture maps) we used; later on, in 3.2, we will explain where and how we used the data assets in the pipeline.

- **colour (Albedo) map** $\mathcal{C}$:

  A colour map is a type of texture map used in 3D modeling and rendering. It defines the base colour of a material and its reflective properties. This map is critical in determining how a material looks under various lighting conditions.

[1] `https://texturing.xyz`

- **Smoothed colour (Albedo) map $\mathcal{A}$**:

  The smoothed version of the colour maps has been constructed from the principal components of the colour maps. Since the principal component stores less frequency information, the reconstructed version of the colour map looks much smoother without fine details.

- **Melanin Map $\mathcal{M}$**:

  The melanin map $\mathcal{M}$, also known as the melanin map, is represented as a three-channel image containing information regarding the melanin and hemoglobin features of the skin. This feature map also comes with a scalar parameter α, which represents the concentration power of the corresponding smooth colour map. The higher the α, the darker the skin colour is going to be.

- **High-Frequency details $\mathcal{H}$**:

  The high-frequency details map $\mathcal{H}$ shows small details on the face, such as wrinkles and beard, and is represented as a single-channel texture map. This map was obtained by applying a Sobel filter [56] to the colour map $\mathcal{C}$.

- **Specular map $\mathcal{S}$**:

  The primary function of a specular map is to control where and how much specular highlights (bright spots that simulate the reflection of light sources) appear on a surface. This is crucial for creating realistic materials and surfaces that interact believably with light. Specular maps allow for fine-tuning the appearance of materials at a granular level. For example, a specular map can make certain parts of a surface look glossy (like a wet road) while other parts remain matte (like dry pavement).

This differentiation is essential for achieving realistic textures and surfaces in 3D environments. Using a specular map is a resource-efficient way to add complexity and realism to a 3D scene without significantly increasing the computational load. Instead of requiring more complex geometry or additional lighting calculations, a specular map modifies how existing lights interact with surfaces.

- **Normal map** $\mathcal{N}$:

  Normal maps give your object texture and depth by changing the direction light is reflected off your 3D model. These are also known as Bump Maps because they make your surface look bumpy. Normal maps don't actually change the geometry of your 3D model, so you can't use them for extreme depth, but they are great for giving realistic textures while keeping the polygon count low.

- **Displacement map** $\mathcal{D}$:

  A displacement map in computer graphics is a texture or image used to alter the geometry of a surface, providing detailed features that enhance the realism of 3D models. Unlike normal maps or bump maps, which only simulate texture detail by manipulating the surface's lighting and appear to modify the surface detail, displacement maps actually change the surface geometry of the 3D model, creating real, tangible depth and detail.

- **Geometry Vertices** V:

  A 13473×3 matrix which holds the values of all the 3D vertices of the mesh, with each vertex having a (x, y, z) location value. It is also worth noting that all the vertices are in correspondence with each other, meaning the n^{th} vertex corresponds

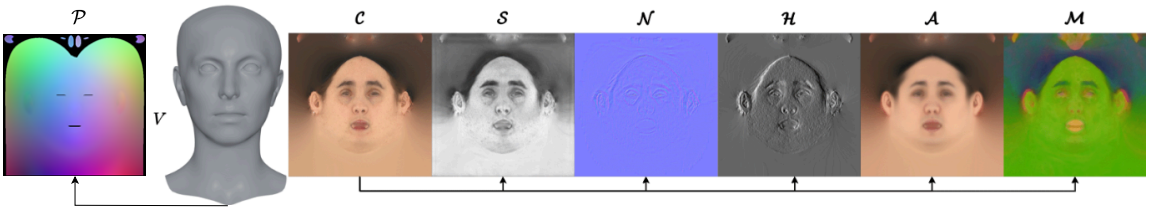


Figure 3.1: A sample from the training set: Position map $\mathcal{P}$, mesh vertices V, colour map $\mathcal{C}$, specular map $\mathcal{S}$, Normal map $\mathcal{N}$, high-frequency map $\mathcal{H}$, smooth colour map $\mathcal{A}$, melanin map $\mathcal{M}$.

to the n^{th} vertex in another mesh. This ensures consistency when using the data. Thus, these vertices form morphable models, not just raw face scans.

- **Position map** $\mathcal{P}$:

  A position map is a texture that encodes the spatial coordinates of points on a 3D surface. Each pixel in the position map corresponds to a point in 3D space, and the colour values of the pixel store the X, Y, and Z coordinates of that point. Using a UV map, we can project a position map onto a 3D geometry. It is worth noting that both Position map $\mathcal{P}$ and Geometry Vertices V represent the same things but in different formats. Position $\mathcal{P}$ in the UV space and Geometry Vertices V in the 3D space.

3.2 Architecture

In this section of the thesis, we will go over each component of the proposed pipeline; in addition, we will also explain the data that was used for the training model and the details regarding its training. As in Figure 3.2, we propose our pipeline: from a Gaus-

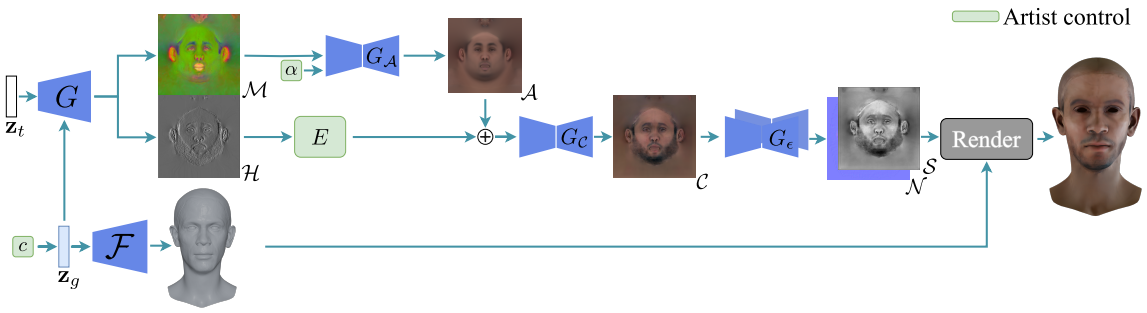


Figure 3.2: Overall Architecture Overview

sian distribution (μ_c, Σ_c), we sample a random vector $\mathbf{z}_g$. By passing $\mathbf{z}_g$ to the Geometry Generator $\mathcal{F}$, a mesh is generated. From $\mathbf{z}_t$ (sampled from a Gaussian distribution), the Geometry-aware Texture Generator G generates two intermediate skin representations: a melanin map $\mathcal{M}$ and a high-frequency details map $\mathcal{H}$. Using the melanin map $\mathcal{M}$ and a scalar α, the Skin colour Control Network $G_{\mathcal{A}}$ generates a smooth colour map $\mathcal{A}$, allowing for precise control of skin colour by adjusting α. By passing $\mathcal{A}$ and $\mathcal{H}$ to the Fine-grained Detail Editing Network $G_{\mathcal{C}}$, a colour map $\mathcal{C}$ is obtained, containing fine details from $\mathcal{H}$. The Face Attributes Decomposition Network $G\epsilon$ decomposes the intrinsic feature maps into a specular map $\mathcal{S}$ and a normal map $\mathcal{N}$, which can be used to render realistic heads. The feature maps obtained thus far are then used to render a face avatar.

3.2.1 Geometry Generator $\mathcal{F}$

For geometry generation, we used a pre-trained part-based Variational Auto-Encoder (VAE) inspired by [2], which was previously developed by the Ubisoft team. The VAE is constructed from 8 independent Graph Neural Networks (GNN) [50]; the encoder of each one of these GNNs is responsible for translating the vertices of a specific part of the face into

a latent representation. These latent representations are then concatenated to each other into a single representation, which we call $\mathbf{z}_g$; afterward, we pass the merged representation $\mathbf{z}_g$ to the decoder of the VAE, which we call $\mathcal{F}$. Decoder $\mathcal{F}$ will reconstruct the entire face mesh. It is also noticeable that the architecture of decoder $\mathcal{F}$ is the same as in [2]. One of the capabilities of these VAE models is that we are able to recover latent representation $\mathbf{z}_g$ from a given mesh via an optimization process. As mentioned in section 3.1, our dataset includes the 3D face geometry. Therefore, we have the ability to recover the latent representation for each face geometry in the dataset. Since we have the label for each of the subjects in terms of gender, ethnicity, and age, we can calculate the statistical parameters of each class, such as mean μ_c and standard deviation σ_c. When it comes to generating new samples from class c, we can sample the latent representation from distribution $\mathbf{z}_g \sim \mathcal{N}(\mu_c, \sigma_c^2)$ to generate face-mesh belonging to a specific group. Unconditioned samples can be generated from a standard Gaussian Distribution $\mathbf{z}_g \sim \mathcal{N}(0, I)$. Figure 3.3 demonstrates that each segment of the face is encoded separately (using different encoders). By combining these encodings, we obtain the $\mathbf{z}_g$, the latent representation. This representation is then passed to the decoder $\mathcal{F}$ to reconstruct a full face mesh.

3.2.2 Geometry-aware Texture Generator

We trained a Geometry-aware Texture Generator G that jointly outputs two intermediate representations for skin texture. These two skin assets are (1) melanin map $\mathcal{M}$ and (2) high-frequency details $\mathcal{H}$. Figure 3.4 illustrates how generator G is being trained. The motivation behind this is that it will separate two controlling aspects of the final face avatar, such as the skin colour and the high-frequency fine-grain details.

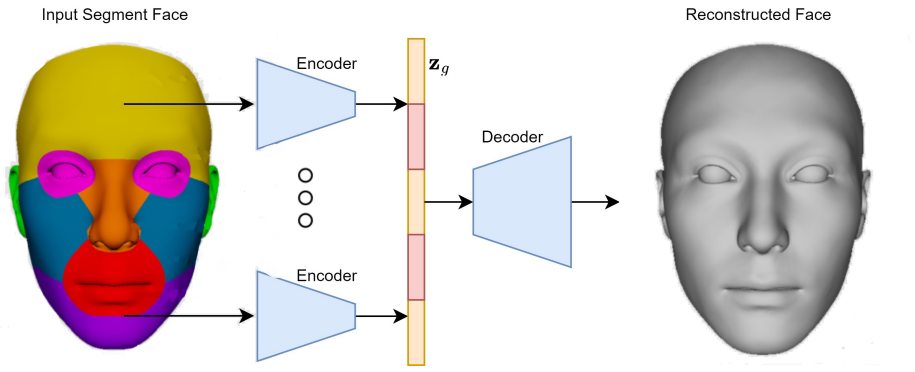


Figure 3.3: Figure from [2] which they feed each segment of the face to its part encoder. Then, they merge and pass the encoded representation to a decoder that reconstructs the face.

In order to encourage network G to learn the correlation between the head geometry and the skin texture, we trained model G conditioned on latent code $\mathbf{z}_g$, which was used for generating a face-mesh via model $\mathcal{F}$. The approach for conditioning was inspired by StyleGAN2 [30], where we concatenate latent code $\mathbf{z}_g$ with a random vector $\mathbf{z}_t$ sampled from a Gaussian distribution $\mathbf{z}_t \sim \mathcal{N}(0, I)$. This will allow us to generate face avatars with similar geometry and different textures by fixing latent code $\mathbf{z}_g$ and changing latent code $\mathbf{z}_t$. During training, we also noticed that using a single discriminator is not sufficient to produce good-quality assets. After many experiments (mentioned in 4.1.3), we noticed the best results were obtained when three distinct discriminators were used. These three discriminators were (i) for melanin map $\mathcal{M}$, (ii) for high-frequency details map $\mathcal{H}$, and (iii) for the combination of both $\mathcal{M}$ and $\mathcal{H}$, to learn the correlation between these two feature maps. The idea for using three distinct discriminators was inspired by [35]. Similar to the original StyleGAN2 [30], each discriminator also takes geometry latent code $\mathbf{z}_g$ as conditioning input. Since G also uses the conditioning latent code $\mathbf{z}_g$ as input to gener-

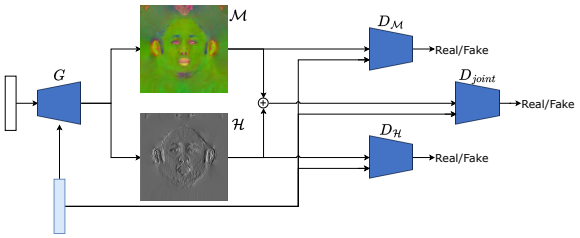


Figure 3.4: Geometry aware texture generator G utilizes $\mathbf{z}_g$ (which was used to generate the head geometry) to generate melanin map $\mathcal{M}$ and high-frequency detail map $\mathcal{H}$.

ate $\mathcal{M}$ and $\mathcal{H}$, similar to the conditioning in 3.2.1 we can sample from $\mathcal{N}(\mu_c, \sigma_c^2)$ (where μ_c and σ_c belong to a specific group) which will result in getting feature maps from that specific group. Equation 3.1 presents the loss function used for training model G. This loss function is inspired by the one used in StyleGAN2 [30], with the key difference being that the adversarial loss in our model is the sum of three adversarial losses from three discriminators. Additionally, we included the Path Length regularization term L_{pl}, as utilized in StyleGAN2, along with the R_1 regularization term [41]. For training this model, we used the same training scheme used in StyleGAN2 [30], with a hyper-parameter $\gamma = 4$. Additionally, we modified the scheme by employing three discriminators instead of one, each contributing equally to the final loss function. The model was trained over approximately 1500 epochs, using a batch size of 8 and a learning rate of 0.002. The input data was normalized to have a mean of 0 and a variance of 1.

$$\begin{aligned} \mathcal{L}_{\text{adv}} &= \mathcal{L}_{\mathcal{M}} + \mathcal{L}_{\mathcal{H}} + \mathcal{L}_{\text{Joint}} \\ \mathcal{L} &= \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{pl}} + \mathcal{L}_{R_1} \end{aligned} \tag{3.1}$$

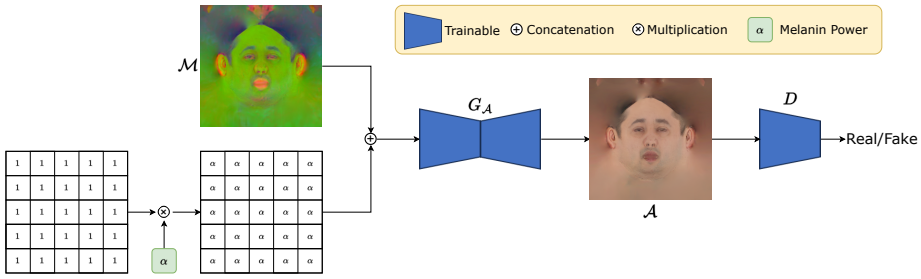


Figure 3.5: Training the skin colour control model $G_{\mathcal{A}}$, involves combining the melanin map, $\mathcal{M}$, with a feature map valued as α. Passing these two as a combination to $G_{\mathcal{A}}$ results in the creation of a smooth colour map $\mathcal{A}$, which is subsequently classified as real or fake by a discriminator.

3.2.3 Skin colour control

For controlling the skin colour of the subject, we trained network $G_{\mathcal{A}}$, which learns a mapping from the melanin map $\mathcal{M}$ and scalar α (which indicates the melanin concentration power) to a corresponding smooth colour map $\mathcal{A}$. Creating an analytical and closed-function relationship between the melanin and hemoglobin space to skin colour space is an extremely difficult and non-trivial task. Due to these limitations, we propose a data-driven solution to tackle this challenge. We constructed a dataset compromising tuples of $\mathcal{M}, \alpha, \mathcal{A}$. As previously mentioned, scalar α represents the melanin concentration, in which lower values of α indicate lighter skin and higher values of α indicate darker skin colour. $\mathcal{A}$ is a smooth texture that discards high-frequency details and keeps the basic details regarding skin colour. More details regarding the datasets are provided in section 3.1. For the following task, an image-to-image translation network [25] was trained, which we denote as $G_{\mathcal{A}}$. In addition to the original implementation inspired from [47], we used the multi-patch, multi-resolution discriminator, which gave us better quality results compared

to the original image-to-image translation network. In order to provide melanin concentration power α as a condition, we created a channel containing α in all positions alongside melanin map $\mathcal{M}$ as the input. The goal was to reconstruct a smooth colour map $\mathcal{A}$. During the training process, we minimized the adversarial loss and the ℓ_2 between the network's output and the corresponding ground truth smoothed colour map $\mathcal{A}$. During inference time, network $G_{\mathcal{A}}$ will encode the melanin map $\mathcal{M}$ generated from G to colour map A. We can adjust the desired skin colour by changing the scalar value α. For training this model, an image-to-image translation network [26] was used; in addition to the original implementation inspired from [47], we used the multi-patch, multi-resolution discriminator where the model was trained for 300 epochs with the batch size of 4. The initial learning rate was 0.0001, and we used the decay scheduling of 0.1 for each 60 epoch. The input data was also normalized to have a mean of 0 and a variance of 1. $G_{\mathcal{A}}$ was trained of 256×256 feature images. Figure 3.5 shows an overview of the pipeline for $G_{\mathcal{A}}$.

3.2.4 Fine-grained detail editing

The final reflectance map $\mathcal{C}$ is obtained when colour map $\mathcal{A}$, which aggregates low-frequency details, and high-frequency details map $\mathcal{H}$, which aggregates high-frequency details, are passed to network $G_{\mathcal{C}}$. Formulating the generation process with these intermediate steps allows simple manipulation of the high-frequency details of the face. An artist can make changes to $\mathcal{H}$ using image-editing software (tools as simple as Microsoft Paint) to add/remove details. To train this network, we create a dataset of tuples of reflectance maps $\mathcal{C}$, high-frequency $\mathcal{H}$, and low-frequency maps $\mathcal{A}$. The same network architecture from $G_{\mathcal{A}}$ has been used for $G_{\mathcal{C}}$, with the exception of the number of input channels and

input dimension. For $G_{\mathcal{C}}$ an image-to-image translation network [26] is used, In addition to the original implementation inspired from [47], we used the multi-patch, multi-resolution discriminator, which gave us better quality results compared to the original image-to-image translation network. The model is trained by taking the concatenation of melanin map $\mathcal{A}$ and high-frequency map $\mathcal{H}$ as the input and learns to recover the final colour map $\mathcal{C}$. During training, we minimize the adversarial loss and the $\ell 2$ distance between the network's output and the corresponding ground truth texture map. For training $G_{\mathcal{C}}$, we employed an image-to-image translation network, as described in [26]. Additionally, inspired by [47], we utilized a multi-patch, multi-resolution discriminator. The model was trained for 300 epochs with a batch size of 2. Inputs, including the smooth colour map $\mathcal{A}$ and high-frequency map $\mathcal{H}$, were upsampled from 256×256 to 512×512 using bi-linear interpolation. For the output, we used a higher-resolution reflectance map $\mathcal{C}$, which did not require upsampling as it was already available in our dataset at the desired resolution. We observed that using a higher resolution produced more finely detailed results compared to the lower-resolution counterpart (256×256). Both of the input data($\mathcal{M}$ and $\mathcal{H}$) were normalized to have a mean of 0 and a variance of 1. Figure 3.6 shows an overview of the pipeline for $G_{\mathcal{C}}$.

3.2.5 Face attributes decomposition

In order to create realistic-looking avatars, modern software renderers such as Blender[2] need more than just colour texture maps. For that particular reason, we used two additional intrinsic maps (specular map $\mathcal{S}$ and normal map $\mathcal{N}$). These two intrinsic maps are

[2] `https://www.blender.org/`

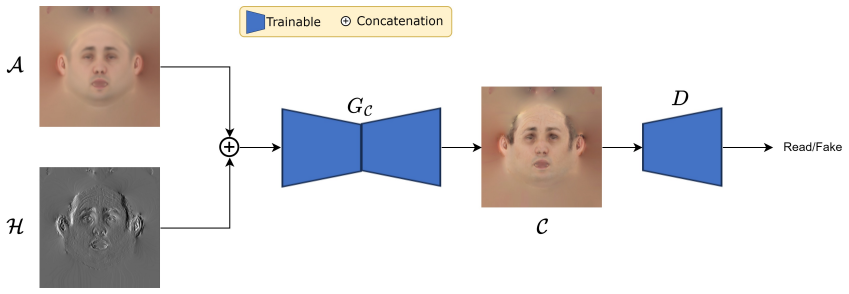


Figure 3.6: Fine-grained detail editing network: Smoothed colour map A is concatenated with high-frequency detail map $\mathcal{H}$ and passed to $G_{\mathcal{C}}$. The result is the creation of colour map $\mathcal{C}$.

estimated from networks G_{ϵ_s} and G_{ϵ_n}. The role of these networks is to ensure that the modifications made by the artist at the colour map level are propagated into other intrinsic feature maps. These maps are essential for realistic rendering, especially under novel lighting conditions, where the difference between using and not using these maps is visible. These two models have been inspired by [10].

3.2.6 Specular Map estimator

Specular Map estimator G_{ϵ_s} was developed by the Ubisoft team. Similar to $G_{\mathcal{A}}$ (in section 3.2.3) and $G_{\mathcal{C}}$ (in sections 3.2.4) an image-to-image translation network [26] with a multi-patch, multi-resolution discriminator which were inspired by [47] was used. For training this model, they used tuples of $\mathcal{C}, \mathcal{S}$, where reflectance map $\mathcal{C}$ serves as the input, while it should return specular map $\mathcal{S}$ as the output. During inference time, the reflectance map generated from $G_{\mathcal{C}}$ is passed to G_{ϵ_s}.

3.2.7 Normal Map estimator

Normal map $\mathcal{N}$ is obtained from the displacement map $\mathcal{D}$ using a Fast Fourier convolution, similar to [35, 64, 10] we trained a network to estimate the displacement map $\mathcal{D}$. Initially, using the super-resolution network explained in section 3.2.8, we upscale colour map $\mathcal{C}$ from 256×256 to 4096×4096. Inspired from [8], two models are responsible for extracting the displacement map. Figure 3.7 shows the Partial Detail Inference Module (PDIM) takes 2D image patches as inputs and generates displacement map PCA coefficients (to gather low-frequency information). Using such a scheme will reduce the parameter space and be more stable during training and testing time. However, PCA-based approximations are not capable of capturing high-frequency information, which is essential for fine detail synthesis. Therefore, we use a model called Partial Detail Refinement Module (PDRM) to refine high-frequency details further. The motivation behind this is to explicitly break down the facial inference procedure into linear approximation and non-linear refinement, which is that facial details consist of both regular patterns like wrinkles and characteristic features such as pores and spots. By using a two-step scheme, we encode such priors into our network. Exactly like [8] the PDIM module, we used a UNet-8 structure concatenated with 4 fully connected layers to learn the mapping from texture map to PCA representation of displacement map. The sizes of the 4 fully connected layers are 2048, 1024, 512, 64. Except for the last fully connected layer. In the subsequent PDRM module, we use UNet-6, i.e., 6 layers of convolution and deconvolution, each utilizing a 4×4 kernel, with a stride size of 2 and a padding size of 1. Apart from this, we adopt the $LeakyReLU$ activation layer for all the layers except the last convolution layer, where for the last layer, we employed a $tanh$ activation function. To train the PDIM and PDRM modules, we integrate

both supervised and unsupervised training techniques, leveraging Conditional Generative Adversarial Networks (cGAN) to effectively manage variations in facial texture.

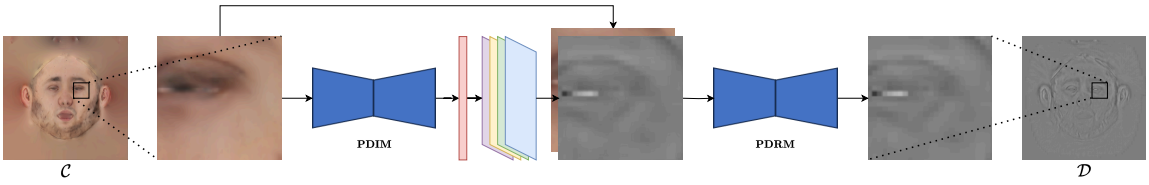


Figure 3.7: Network architecture for facial detail synthesis. PDIM for medium-frequency detail (wrinkles) synthesis and PDRM for high-frequency detail (pores) synthesis

3.2.8 Super-Resolution Network

In order to achieve detailed results, all of the generated feature maps ($\mathcal{C}$, $\mathcal{S}$) are up-sampled using two super-resolution networks. The architecture of these two models was inspired by Enhanced Super-Resolution Generative Adversarial Network (ESRGAN) [63], which were trained by other Ubisoft team members. Similar to [35], the process of upscaling is done in two separate stages, where in the initial stage, the feature maps are upscaled from 256×256 to 1024×1024 and in the second step, the feature maps are upscaled from 1024×1024 to 4096×4096. For training the first model (256×256 to 1024×1024), tuples of $\{\mathcal{C}_{256\times256}, \mathcal{C}_{1024\times1024}\}$ and $\{\mathcal{S}_{256\times256}, \mathcal{S}_{1024\times1024}\}$ were used (both data types were used for a single model), and for the second model (1024×1024 to 4096×4096) tuples of $\{\mathcal{C}_{1024\times1024}, \mathcal{C}_{4096\times4096}\}$ and $\{\mathcal{S}_{1024\times1024}, \mathcal{S}_{4096\times4096}\}$ were used (both data types were used for a single model).

Chapter 4

Results

In this section, we will go over each one of the different experiments that have been done and explain the motivation behind the experiment alongside our findings and results from that experiment.

4.1 Geometry and texture generation

In this experiment, we evaluate two aspects of geometry and texture generation: (i) comparing our GNN generator operating on vertices with a CNN generator operating on UV space, and (ii) evaluating the impact of conditioning our texture generation model with geometry.

To highlight the advantages of using a Graph Neural Network (GNN) for geometry generation, we compare our method to a StyleGAN-based generator that jointly generates texture and geometry in UV space (similar to [35]), as the **Baseline** (see figure 4.1).

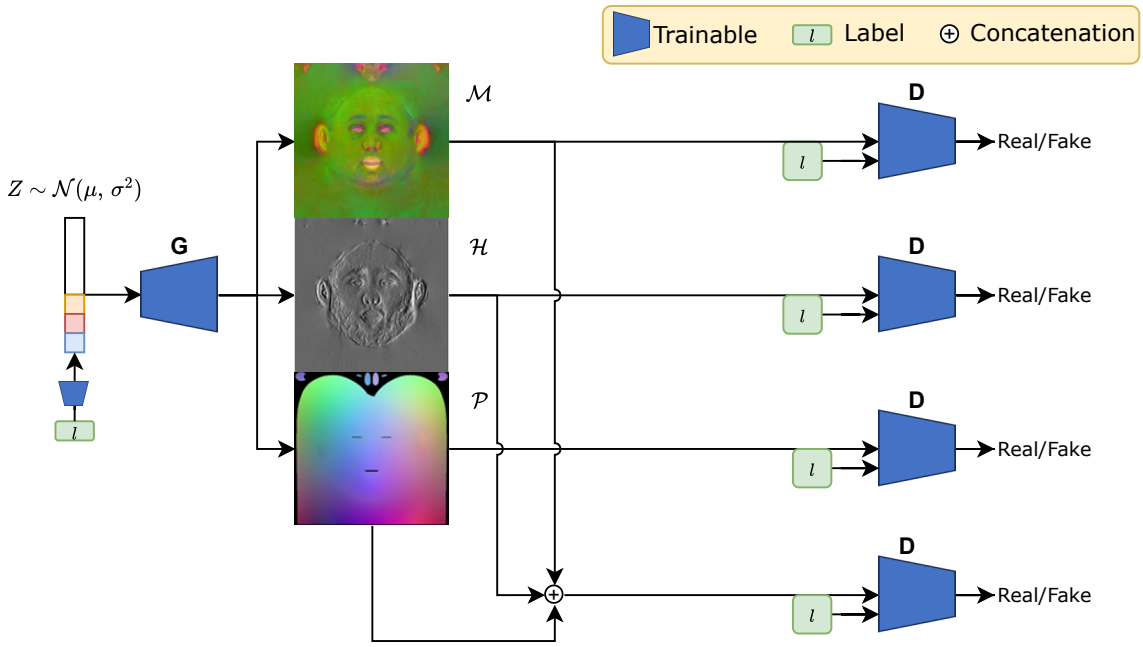


Figure 4.1: Baseline: From G melanin map $\mathcal{M}$, high-frequency map $\mathcal{H}$, and position map $\mathcal{P}$ are generated. A discriminator is used for each feature map, and a single discriminator is used for a concatenation of all of the feature maps.

4.1.1 Baseline

Pioneering works such as [35, 39] have used a single generative model to generate colour maps and position maps, which were then used to create a face rendering. These works inspired us to use the same approach to generate melanin map $\mathcal{M}$, high-frequency map $\mathcal{H}$, and position map $\mathcal{P}$ (which were then projected to 3D vertice space V using the UV map). We will use this approach as a **Basline**.

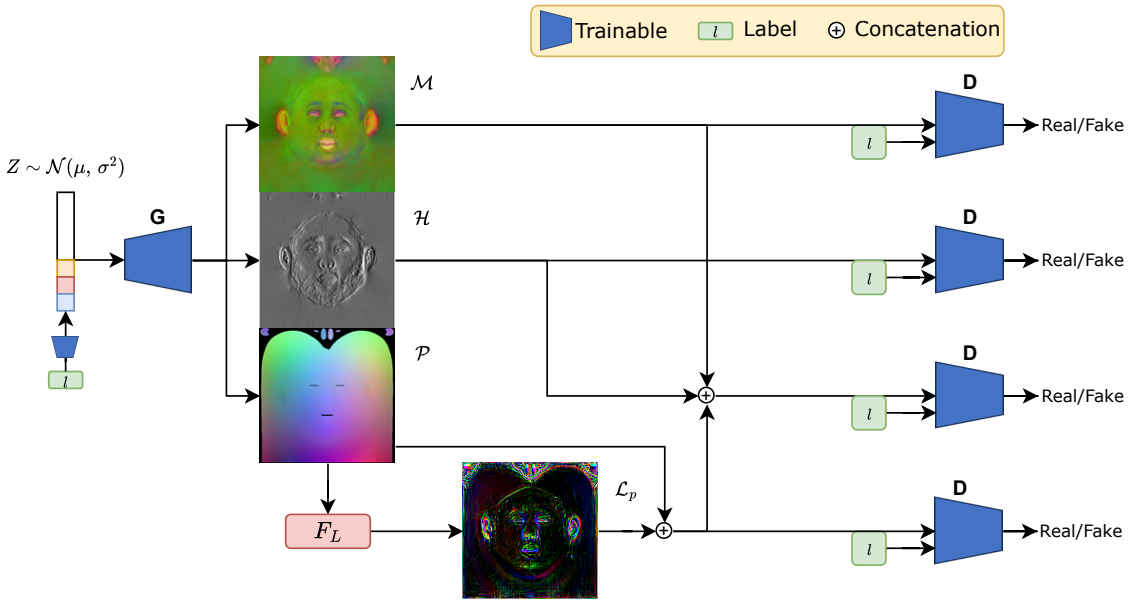


Figure 4.2: Baseline Laplacian 3D: In addition to the Baseline where the melanin map $\mathcal{M}$, high-frequency map $\mathcal{H}$, and position map $\mathcal{P}$ are generated, a Laplacian filter is applied to the position map, resulting in new features that are used as additional signals for the discriminators.

4.1.2 Baseline Laplacian 3D

In addition to the baseline, we developed a novel approach that improved the quality of the rendered faces and increased the convergence speed. When using the baseline, we noticed small artifacts on the face mesh that were not apparent in a normal render. These small artifacts can be seen when we applied a Laplacian filter [18] on position map $\mathcal{P}$. To solve this, we propose a novel approach where we use the Laplacian position map as an additional signal to train our generator. As in Figure 4.2, we denote the laplacian of the position map $\mathcal{L}_p$. When $\mathcal{L}_p$ is used as an additional signal, we notice higher quality results (FID score can be seen in table 4.1) and faster convergence. More details regarding the implementation can be found in the appendix A.2.

4.1.3 Quantitative Comparision of GNN and Baseline

To evaluate our model, we began by randomly selecting 10,000 pairs of latent vectors, denoted as $\mathbf{z}_g$ and $\mathbf{z}_t$ (from Gaussian distribution). Using $\mathcal{F}$, we generated geometry vertices V from the latent vectors $\mathbf{z}_g$. Both $\mathbf{z}_g$ and $\mathbf{z}_t$ are passed to the Geometry-aware Texture Generation G to produce the melanin map $\mathcal{M}$ and the high-frequency detail map $\mathcal{H}$. Following this, we applied $G_{\mathcal{A}}$ to the generated $\mathcal{M}$ with a randomly chosen melanin intensity α, yielding a smooth colour map $\mathcal{A}$. Finally, we passed smooth colour map $\mathcal{A}$ and $\mathcal{H}$ to network $G_{\mathcal{C}}$ to get the detailed colour map $\mathcal{C}$. Given colour map $\mathcal{C}$ and geometry vertices V, we render 10,000 images of faces (due to complexity and time constraints, we did not do the upsampling to the colour maps). Figure 4.3 illustrates some examples of these renders.

For the evaluation of our model, the Fréchet Inception Distance (FID) was calculated across four different scenarios: (1) rendered images from V_i and $\mathcal{C}_i$, where V_i represents the geometry vertices of the i^{th} subject and $\mathcal{C}_i$ is the reflectance map of the same subject, generated through our pipeline (referred to as "Ours" in table 4.1). (2) rendered images from V_i and $\mathcal{C}_j$, where V_i and $\mathcal{C}_j$ are the geometry vertices and reflectance map of the i^{th} and j^{th} subjects, respectively, chosen randomly from the 10,000 samples (this scenario is dubbed "Ours un-conditional" in table 4.1). (3) rendered images utilizing V and $\mathcal{C}$ where V was derived from the position map $\mathcal{P}$ generated by the Baseline using Laplacian, and $\mathcal{C}$ was produced from $\mathcal{A}$ (obtained from $\mathcal{M}$) and $\mathcal{H}$, both of which were generated by the Baseline with Laplacian (this setup is detailed in section 4.1.2). (4) a similar approach to scenario 3, but employing only the Baseline without the Laplacian enhancement (outlined in section 4.1.1). For the ground truth data, we used all of the colour maps $\mathcal{C}$ and geometry

Figure 4.3: Top: Samples generated from our model (using GNN). Bottom: Samples generated from the baseline (using a StyleGAN as the generator for geometry)

V to create 892 face renders.

We utilized the rendered images to compute the Fréchet Inception Distance (FID) [21], comparing the distribution of our generated images to that of real images, based on the features extracted by an Inception-v3 model [57]. The performance of our GNN-based model notably exceeded that of the baseline method, suggesting that the distribution of our generated heads is closer to that of the training data. We observed that conditioning the geometry-aware texture generator G on $\mathbf{z}_g$ (thus aligning vertices V and the reflectance map $\mathcal{C}$) slightly improved accuracy compared to unconditioned operation, where V and $\mathcal{C}$ were randomly paired. Additionally, incorporating the Laplacian of the position map as an extra signal resulted in significantly enhanced FID scores, underscoring the effectiveness of this method in producing high-quality generated heads.

Method	FID ↓
Ours	**11.44**
Ours un-conditioned	11.53
Baseline+Laplacian	15.35
Baseline	21.28

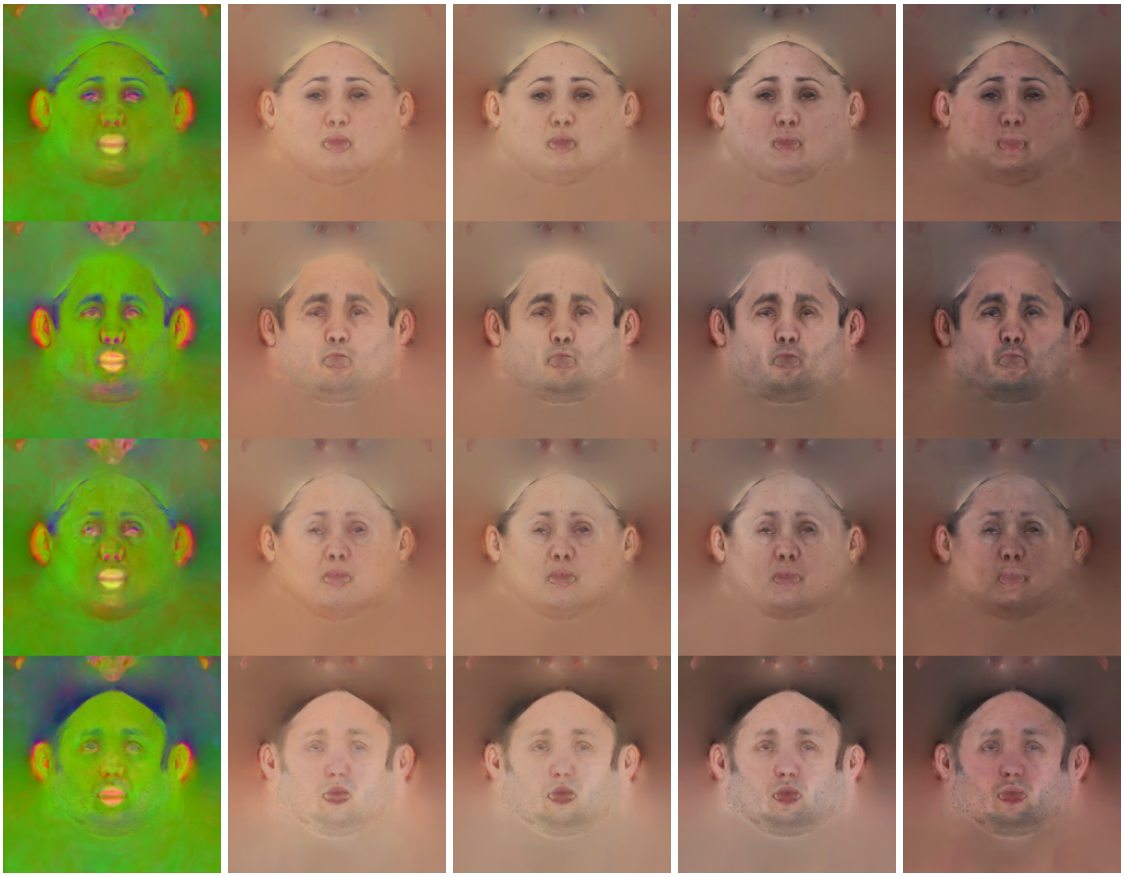
Table 4.1: Quantitative comparison between our method and the baseline.

4.2 Skin colour manipulation

Our pipeline enables precise skin colour manipulation with a single scalar melanin power α. In this experiment, we show precisely how scalar α will affect the colour map and face render. Figure 4.4 displays how α will affect the colour map $\mathcal{C}$. In addition, figure 4.6 shows the effect of different α for different subjects. We also did an experiment where we visualized the change in shape and colour independently in the same figure, where figure 4.5 shows that linear interpolation $\mathbf{z}_g$ and melanin power α will change the appearance of the subject. However, we should point out that one of our limitations is that high value α may lead to small unwanted details(e.g., beard) or artifacts. For example, in figures 4.4, 4.5, 4.6, the beard becomes visible for darker skin characters. We assume this is due to the dataset that was used during training, where darker-skinned subjects tend to have more facial hair compared to lighter-skinned characters.

4.3 Evaluating Skin colour Control

In an experiment that other members at Ubisoft did to quantitatively evaluate the precision of our skin colour control method, they compared our method to a baseline that consists of manipulating the hue-saturation-value (HSV) of the texture, which is a common approach



(a) $\mathcal{M}$ (b) $\mathcal{A}_{\alpha=1}$ (c) $\mathcal{A}_{\alpha=1.4}$ (d) $\mathcal{A}_{\alpha=1.8}$ (e) $\mathcal{A}_{\alpha=2.2}$

Figure 4.4: How Melanin power (α) affects the colour tone for four different subjects.

used by artists for skin colour editing. We hypothesize that our method is better at matching the lip colour for a given skin colour. We designed an experiment to evaluate this, as follows: using our pipeline, 1000 head samples are generated. For each melanin map, we produce two textures with two different values of α, namely source $\mathcal{C}_{src}$ and target $\mathcal{C}_{tgt}$. We use an optimization procedure to find the optimal HSV value that, added to $\mathcal{C}_{\text{src}}$, matches the target skin colour $\mathcal{C}_{\text{tgt}}$. We denote the optimized resulting texture $\mathcal{C}_{\text{hsv}}$.

Given $\mathcal{C}_{\text{tgt}}$ and $\mathcal{C}_{\text{hsv}}$, we find the 5 closest textures in the training dataset in terms of

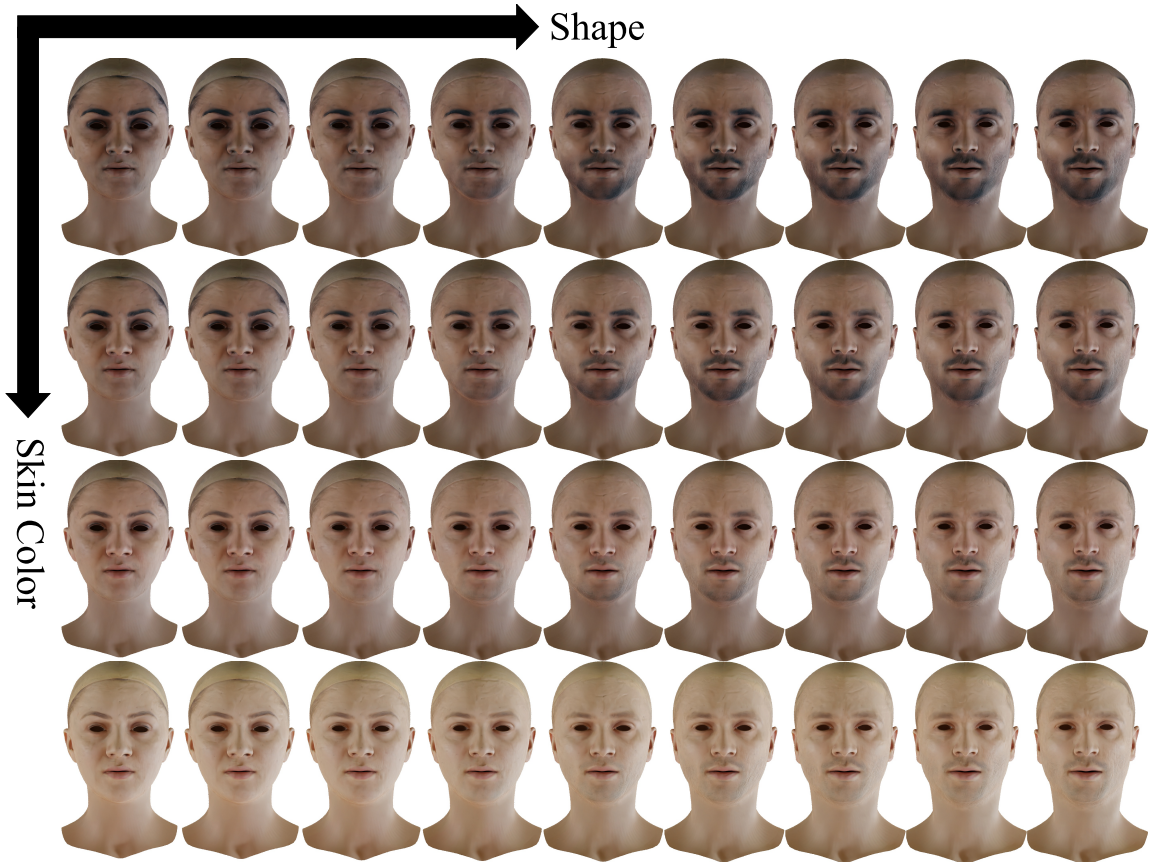


Figure 4.5: Left to right: The change of the latent representation $\mathbf{z}_g$ resulting in the change of face geometry and melanin map $\mathcal{M}$. Top to bottom: The change of melanin power α resulting in different face skin colour.

skin colour using the Individual Typology Angle (ITA) distance [7, 13, 40], that measures skin colour with a single scalar. Finally, we compute the error mean and standard deviation on the Hue component between the generated textures and their corresponding closest textures in the dataset on the lips region. Table 4.2 reports the average error and standard deviation on the Hue component for our method and the HSV method. Our method demonstrates significantly lower error compared to the HSV method and exhibits significantly lower variation. This shows that our model is better than the HSV editing at

Figure 4.6: Melanin power α shift for different subjects

correlating lip colour with skin colour. Figure 4.7 shows samples of renderings of both methods (HSV and ours). We noticed that the HSV approach occasionally obtains an unrealistic, greenish colour for the lips, while our method produces natural lips colour matching the face skin colour. Moreover, the HSV-based linear model tends to generate reddish colours for dark skin colour subjects, while our model produces a more realistic rendering.

Method	Mean ↓	STD ↓
HSV	0.295	0.235
Ours	**0.252**	**0.116**

Table 4.2: Error on the lips region of our method compared to HSV skin tone editing.

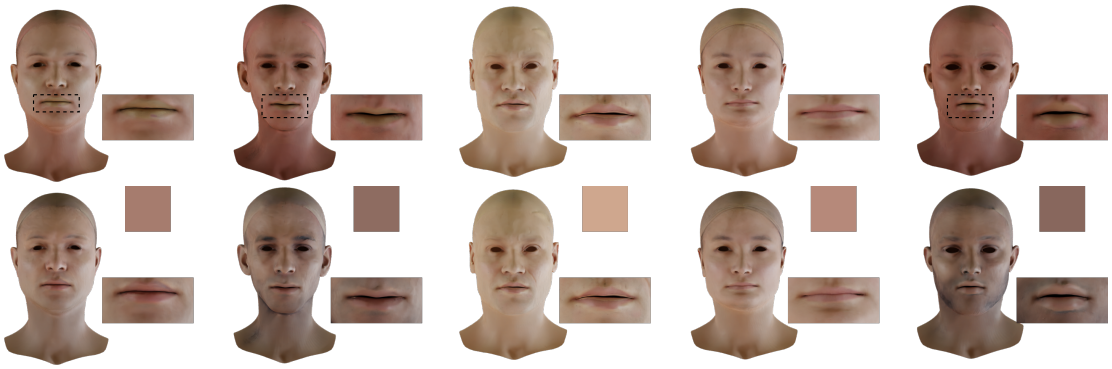


Figure 4.7: Results on skin colour editing with HSV (top) and Ours (bottom).

4.4 Fine-grained details manipulation

So far, by manipulating the latent codes $\mathbf{z}_g$ and $\mathbf{z}_t$, we have controlled the subject's face geometry and general appearance. As detailed in section 4.2, adjusting the melanin factor α changes the skin colour of subjects, providing a basic method for modifying skin colour attributes. However, these methods do not incorporate or eliminate detailed characteristics.

Our pipeline allows for manipulations of fine-grained face details using a single-channel map $\mathcal{H}$ while preserving the subject's skin colour. Artists can make arbitrary changes to this map using standard editing tools (e.g., Microsoft Paint), and these changes are coherently propagated to the intrinsic facial feature maps. To validate this property, we asked an artist to make modifications for two subjects generated by the model. In the first task, the artist added wrinkles to make a character look older. In the second task, the artist removed the beard of a subject, which is a common task required to clean up scans since beards

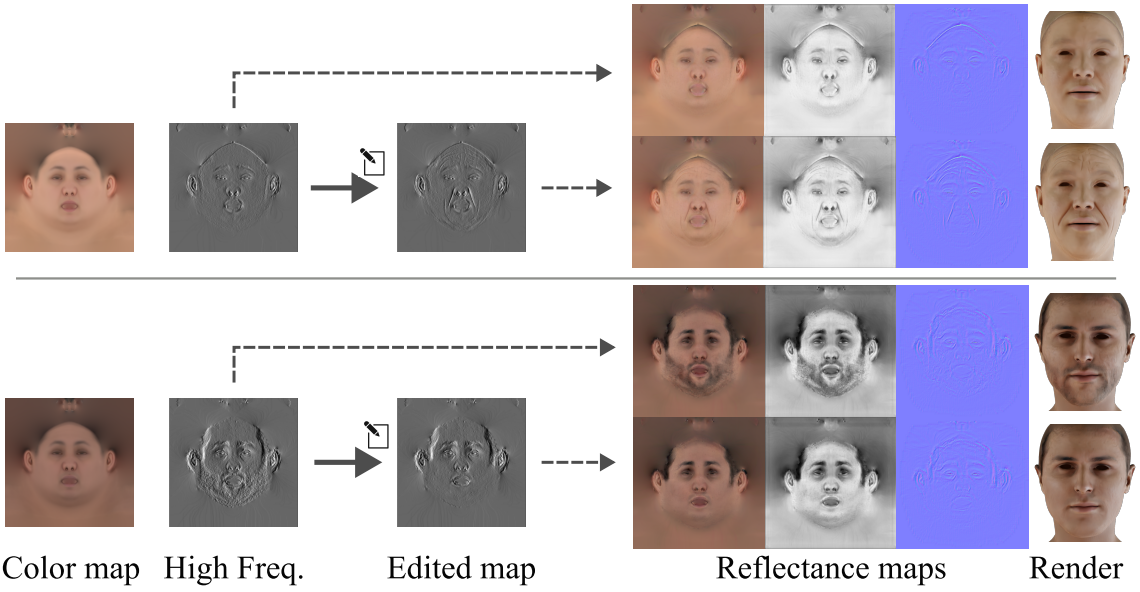


Figure 4.8: Example of artistic editing of high-frequency details. We show an example of adding wrinkles (top subject) and removing a beard (bottom subject). Changes in the High-frequency detail map $\mathcal{H}$ are cohesively propagated to the intrinsic facial maps (colour, specular, and normal).

are generally modeled separately from the skin in rendering engines. Figure 4.8 shows the results of this process. We observed that changes to the fine-grained details map $\mathcal{H}$ are properly propagated to all intrinsic facial maps (colour, specular, and normal) while preserving the original skin colour of the subject, thus simplifying the process of making these changes. Figure 4.8 shows additional experiments conducted on edge editing. Figure 4.9, we show additional samples on how the high-frequency detail map $\mathcal{H}$ will result in different-looking renders.

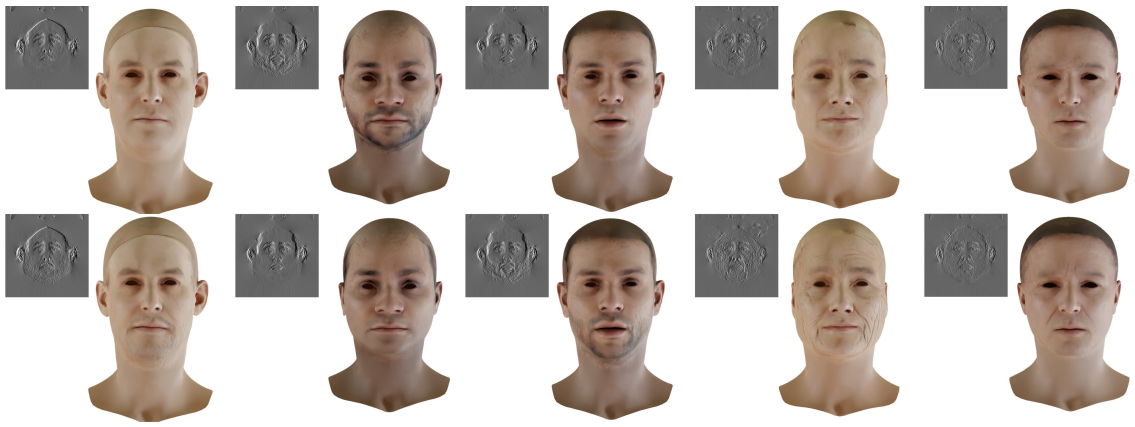


Figure 4.9: Additional examples of high-frequency detail editing. Top: Render of the original asset. Bottom: Render after changes to the map $\mathcal{H}$.

4.5 Skin control editing for *in-the-wild* portrait images

In this section, we demonstrate our method's capability to adjust the skin colour of subjects by manipulating existing texture maps obtained from real-world 2D portrait images. We trained a network, $G_{\mathcal{M}}$, to convert from the colour space to the melanin-hemoglobin space (exactly the opposite of $G_{\mathcal{A}}$. Similar to $G_{\mathcal{A}}$ (in Section 3.2.3), $G_{\mathcal{C}}$ (in Sections 3.2.4), and G_{ϵ_s}, we utilized an image-to-image translation network [26] equipped with a multi-patch, multi-resolution discriminator, inspired by [47]. For training $G_{\mathcal{M}}$, we employed tuples of $\{\mathcal{M}, \mathcal{C}\}$, where the colour map $\mathcal{C}$ serves as the input, and the network is expected to return the melanin map $\mathcal{M}$ as the output. For the experiment, we utilized a reconstruction model [10] to extract the colour map $\mathcal{C}_m$ from a portrait image. The extracted colour map is then passed to $G_{\mathcal{M}}$, which produces $\mathcal{M}_m$. By feeding the generated melanin map $\mathcal{M}_m$ along with a scalar α into $G_{\mathcal{A}}$ (Section 3.2.3), we can generate colour maps with different skin colour (but same identity). The resulting colour maps were then used for rendering. Figure 4.10 showcases the rendering of the subject with varying melanin power

α. This experiment shows that our method can be seamlessly integrated with 3D face reconstruction methods, providing additional control for artists working with real-world images.

4.6 Effect of using a super-resolution network

In section 3.2.8, the necessity of integrating a super-resolution network into our pipeline is discussed. The primary goal is to create realistic-looking avatars; therefore, the generation of fine-grained details, even if they are technically hallucinated (but look real), is acceptable as long as the end result appears realistic. Figure 4.11 illustrates the impact of employing a super-resolution network to upscale the colour map $\mathcal{C}$, in contrast to scenarios where no upscaling is applied. This enhancement brings out detailed features, which are particularly noticeable around the eyebrows and chin of the subject.

4.7 Comparison against AlbedoGAN

For generative tasks, comparison has always been a challenging task. Some of the well-known metrics are known as FID (section 2.2.1) and KID (section 2.2.2). However, as explained, these metrics measure the distance between the distribution of the ground truth and generated images, which only makes sense when comparing two models that are trained on a similar dataset. Since our pipeline was trained on an internal dataset belonging to Ubisoft, we can't quantitatively compare our approach with other existing works. However, we were able to get some samples from AlbedoGAN [49], thanks to the help of their authors. As a result, we made a visual comparison between our work and Albedo-

GAN, in which we mentioned our advantages. Therefore, in this section, we will discuss the comparison between our method with those from AlbedoGAN. AlbedoGAN uses a pre-trained StyleGAN [28] model to generate a 2D image, followed by the reconstruction of the geometry (using a 3D morphable model), colour map, and normal. In figure 4.12, we show a few samples provided from the AlbedoGAN (left) and our samples (right). For each sample, we show the final render (frontal and side view) along with the corresponding estimate feature maps. Here are a few differences between AlbedoGAN and our work:

- AlbedoGAN does not estimate a complete texture, whereas our method does. Specifically, the side view shows blurry texture for AlbedoGAN and exhibits misalignment between the geometry and the estimated texture, particularly noticeable around the mouth and eyes region. In contrast, our method produces the correct alignment of the geometry and texture (green rectangle in Figure 4.12).

- For AlbedoGAN, certain wrinkles appear in the colour map but are absent in the normal map (this implies that these features only exist on the colour map level and not on the geometry). In contrast, our method produces consistent maps where the wrinkles and effects are visible on all feature maps (blue rectangle in Figure 4.12).

- AlbedoGAN incorporates lighting, specular, and shading information into the generated colour map, whereas our method produces disentangled feature maps, making it more suitable for relighting than AlbedoGAN. On the colour maps from AlbedoGAN, we can see some parts have this lighting effect, which seems brighter than the other parts.

	data-data	generated-data
L2 distance (Geometry)	0.38	0.35
L2 distance (Melanin-hemoglobin)	0.43	0.45

Table 4.3: The average L2 distance between closest samples in the dataset (data-data), and between generated samples and their closest match in the data (generated-data).

4.8 Evaluation of Model Generalization and Diversity

To complete our analysis, we aimed to evaluate the capacity of our geometry generator $\mathcal{F}$ and geometry-aware texture generator G, specifically its ability to craft new samples rather than merely replicating the training data. Figure 4.13 displays generated samples alongside their nearest counterparts from the dataset, illustrating that our method generates heads that are distinctively different from the training dataset's content.

For a quantitative evaluation of our generator's ability to generalize, we conducted the following experiment: We utilized our pipeline to generate a set of 10,000 samples. For each generated mesh V and melanin map $\mathcal{M}$, we identified the closest sample within the dataset using the $\ell 2$ distance and calculated the average of these distances or the entire 10,000 samples. For comparison, we also calculated this metric for samples within the dataset and their nearest matches in the same dataset (excluding itself). The outcomes, presented in Table 4.3, reveal a comparable distance between the closest pairs within the dataset and the closest pairs between the generated samples and the dataset. This indicates that our model is capable of generating a diverse range of samples, confirming its generalization capacity.

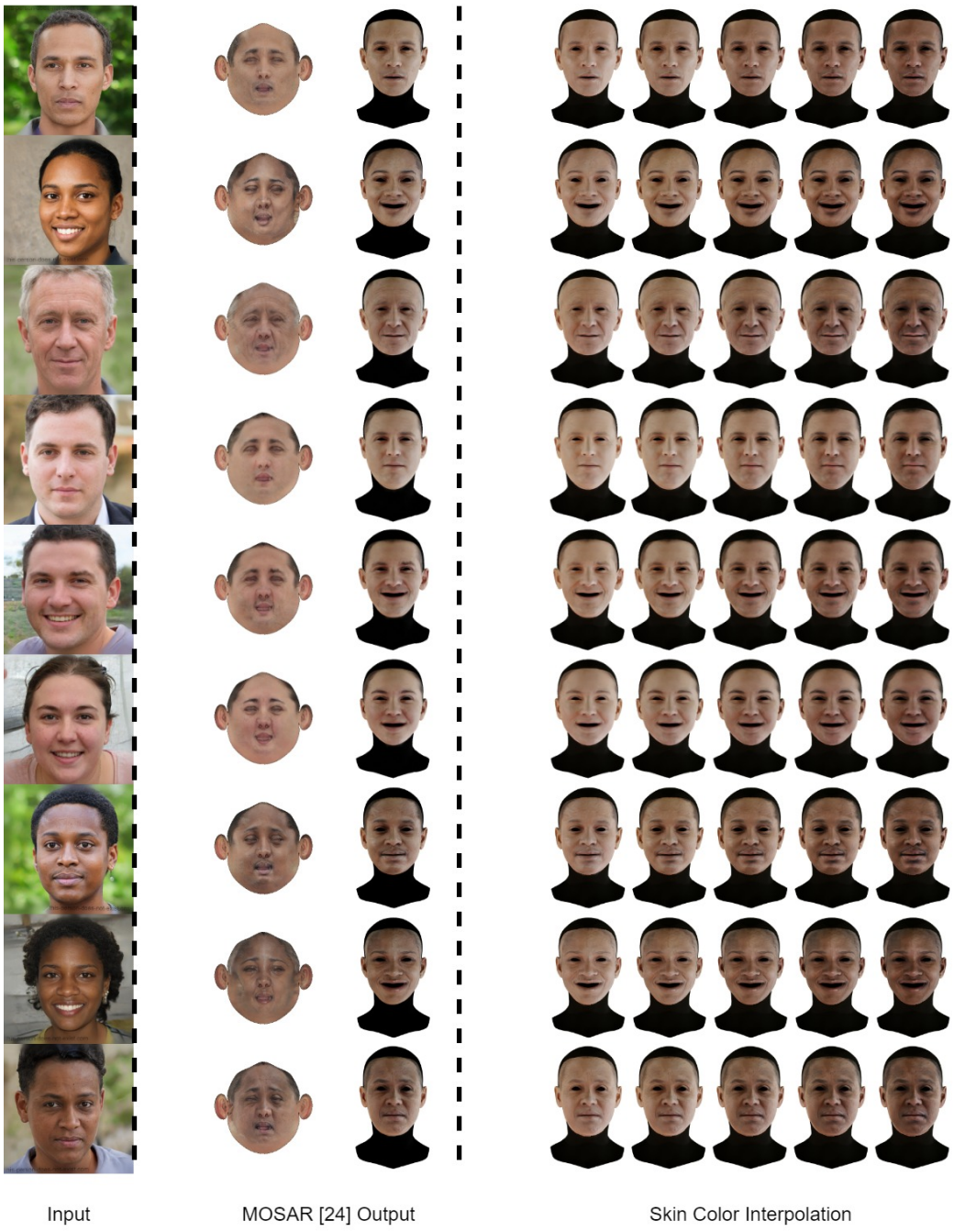
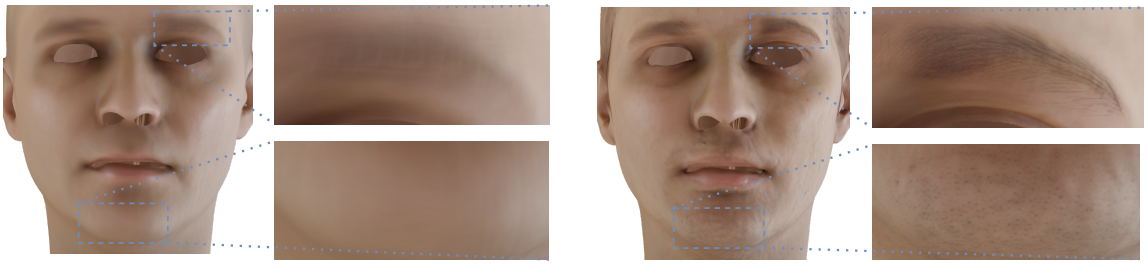


Figure 4.10: Our method allows skin colour control for *in-the-wild* 2D portrait images. First column: Input image. Second and third column: Original texture and render using existing 3D reconstruction method [10]. Remaining columns: Rending with different skin colours obtained by our method.



(a) With super-resolution network

(b) Without super-resolution network

Figure 4.11: Effect of using the super-resolution network to upsample the colour map compared to bilinear upsampling.

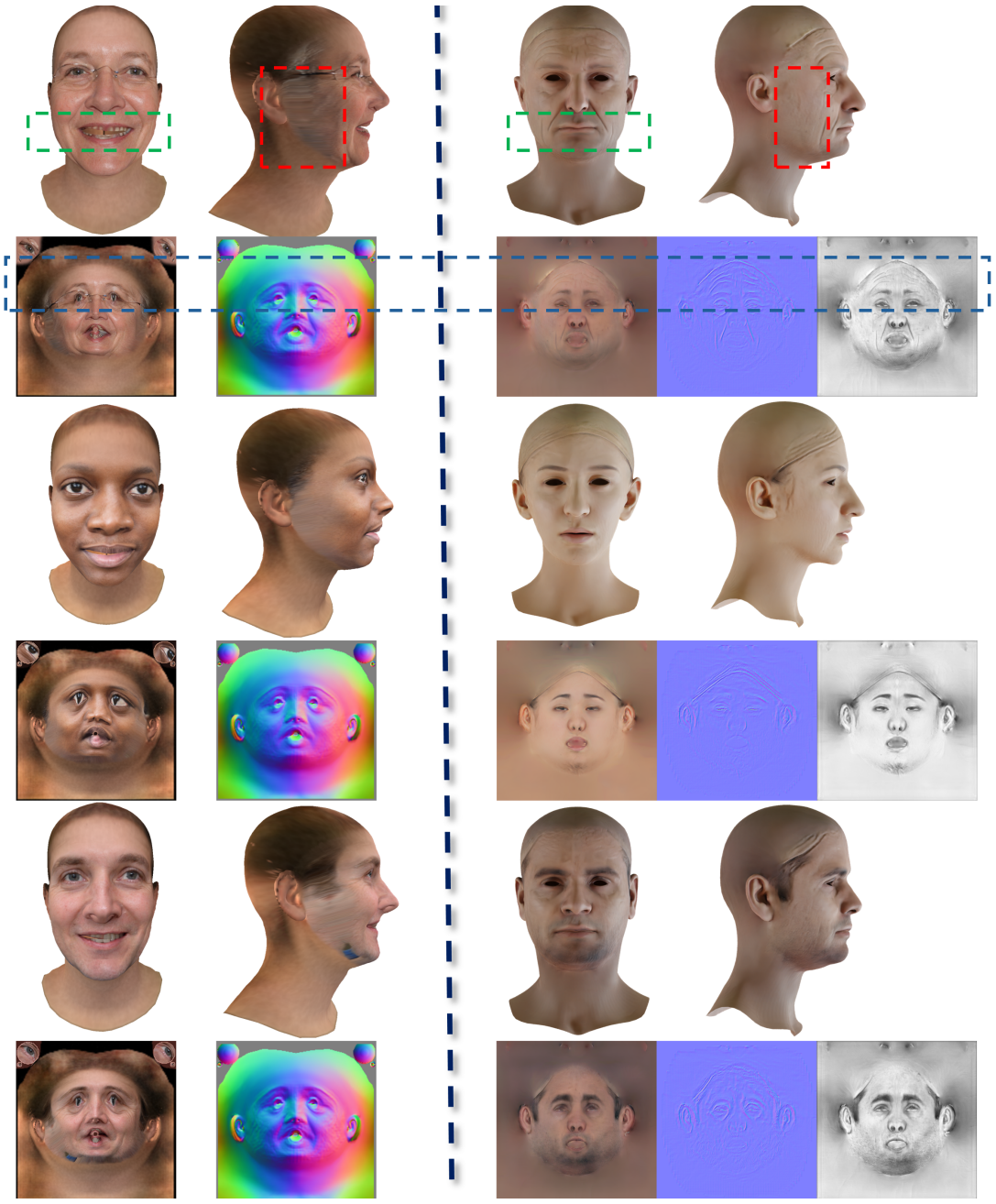


Figure 4.12: Samples from AlbedoGAN [49] (left) and our method (right). For each method, we show (1) frontal and side view rendering and (2) generated texture maps. AlbedoGAn generates a colour map and a normal map (in object space). Our method generates colour, normal (in tangent space), and specular maps.

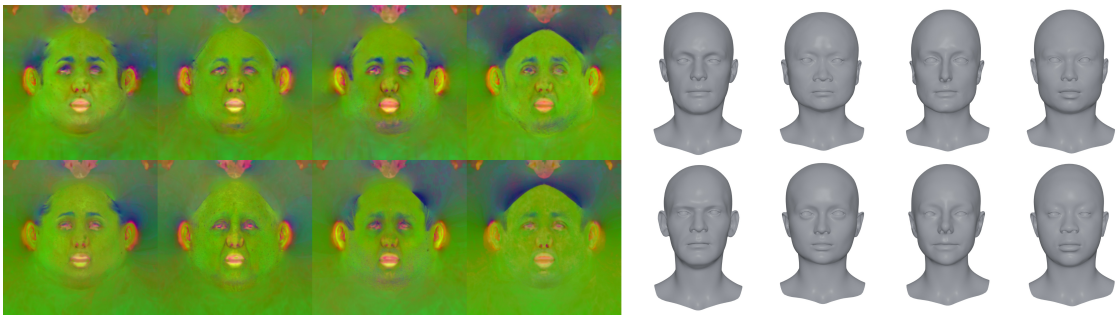


Figure 4.13: Top: Randomly generated samples from our model. Bottom: Closest sample in the dataset to the generated sample.

Chapter 5

Discussion

In this section, we review the main findings and conclusions of this thesis. Although the results are encouraging, it is important to recognize the limitations that could have affected our outcomes, and we identify these limitations identify potential areas for future research in this field.

We introduced an innovative framework for AI-assisted creation of head avatars, offering detailed artistic control at multiple levels. This method enhances the speed and quality of producing realistic-looking heads through different stages. Where in the GNN-based model, we showed that we could create diverse-looking face geometry; additionally, we can control different segments of the face geometry by editing the latent code $\mathbf{z}_g$ (explained in section 3.2.2 and research paper [2]). Our technique is demonstrated to generate reasonable face avatars without the need for post-production adjustments to the geometry meshes ([35] apply post-processing). Geometry-aware texture generator process, merging the strengths of GNN-based models for mesh editing with CNN-based models for texture

refinement, which allowed us to control the feature maps of the face avatar. Our pipeline also consisted of networks that gave precise control over skin colour. It includes a skin colour alteration feature that permits users to adjust skin colour without affecting too many other facial characteristics or causing distortions in non-skin areas (such as the lips). With this, we can solve some potential biases against less-represented ethnic groups in training datasets making sure that we are not limited to producing subjects with a specific skin. Moreover, our strategy for modifying high-frequency details significantly simplifies texture editing by ensuring edits on a single high-frequency detail map are propagated across all other feature maps. This feature proves particularly beneficial for scan cleanups (e.g., beard or wrinkle removal) and for aging or de-aging the 3D avatars. It could also be concluded that our pipeline provides more precise control over the final face avatar compared to the other works mentioned in section 2.3.

5.1 Limitations

Although our pipeline successfully generates high-quality facial avatars, it does show some limitations:

- Regarding the generation of face geometry V, as discussed in section 3.2.1, we can sample from the distribution $\mathcal{N}(\mu_c, \sigma_c)$ to generate a face from class c. However, for classes that are less represented in the dataset, it is challenging to determine whether the model effectively produces accurate face from class c.

- The skin colour control network $G_{\mathcal{A}}$ often struggles with generating dark skin subjects (high values of α). This issue stems from the limited representation of darker-

skinned individuals in our dataset, resulting in skin colour maps that appear blurry and greyish when α is significantly high.

- The fine-grained details editing network $G_{\mathcal{C}}$, which is designed to add fine details to the colour map $\mathcal{C}$, tends to show bias by generating more beards for darker-skinned characters. Additionally, for extremely dark subjects, artifacts around the nose and mouth are sometimes observed (see figure 4.6).

5.2 Future Work

This thesis has explored the capabilities and limitations of 3D face avatar generation, opening doors for future research and development. While the results achieved were promising, they also highlight areas where further enhancements and explorations can be done. The following potential research directions aim to expand the scope of application and improve the methodologies used within this field.

- **Improving Class-Specific Generation Accuracy:** One primary goal is to refine the model's ability to condition the generated face on specific demographic groups more accurately. The current system uses the Gaussian distribution sampling technique for generating a subject, which may not yield precise results, especially for classes that are underrepresented in the training dataset. Future work could develop more robust methods to handle diverse data samples, resulting in better accuracy across different demographic groups.

- **Enhancing Controllability:** Our pipeline introduces three stages of controllability in the generation process. In future work, we can expand this by providing additional

controlling tools to the artists, which would allow for more fine and detailed editing of the face, possibly through enhanced interface designs or more precise control.

- **Refinement of Feature Maps:** The existing system could be extended to include better management of feature maps. Our pipeline is currently generating feature maps, which are required for modern renderers. We can assume that in the future, more advanced renderers are going to be available, and for them, we would require more advanced feature maps. In future work, we can use the same approach to generate those feature maps.

- **Dynamic Expression Modeling:** Currently, the system generates static faces with neutral expressions. Integrating a model to handle dynamic expressions would be a significant advancement, allowing the generation of avatars that can display a range of emotions, which is particularly valuable for animations and interactive applications.

Bibliography

[1] Rameen Abdal, Peihao Zhu, Niloy J Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (ToG)*, 2021. 23

[2] Mohammad Amin Aliari, Andre Beauchamp, Tiberiu Popa, and Eric Paquette. Face editing using part-based optimization of the latent space. In *Computer Graphics Forum*. Wiley Online Library, 2023. viii, 22, 23, 31, 32, 33, 60, 78

[3] R Rox Anderson and John A Parrish. The optics of human skin. *Journal of Investigative Dermatology*, 1981. 25

[4] Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of the 26th annual conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., 1999. 1, 22, 24

[5] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 2014. doi: 10.1109/TVCG.2013.249. 6

[6] Zenghao Chai, Haoxian Zhang, Jing Ren, Di Kang, Zhengzhuo Xu, Xuefei Zhe,

Chun Yuan, and Linchao Bao. Realy: Rethinking the evaluation of 3d face reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 22

[7] Alain Chardon, Isabelle Cretois, and Colette Hourseau. Skin colour typology and suntanning pathways. *International Journal of Cosmetic Science*, 1991. 48

[8] Anpei Chen, Zhang Chen, Guli Zhang, Kenny Mitchell, and Jingyi Yu. Photo-realistic facial details synthesis from single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 39

[9] Ugur Demir and Gözde B. Ünal. Patch-based image inpainting with generative adversarial networks. abs/1803.07422, 2018. URL `https://api.semanticscholar.org/CorpusID:4025915`. 17

[10] Abdallah Dib, Luiz Gustavo Hafemann, Emeline Got, Trevor Anderson, Amin Fadaeinejad, Rafale M.O Cruz, and Marc-André Carbonneau. Mosar: Monocular semi-supervised model for avatar reconstruction using differentiable shading. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. ii, x, 38, 39, 52, 56, 79

[11] Craig Donner, Tim Weyrich, Eugene d'Eon, Ravi Ramamoorthi, and Szymon Rusinkiewicz. A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM Transactions on Graphics (TOG)*, 2008. 25

[12] Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami

Romdhani, et al. 3d morphable face models-past, present, and future. *ACM Transactions on Graphics (TOG)*, 2020. 1

[13] Haiwen Feng, Timo Bolkart, Joachim Tesch, Michael J Black, and Victoria Abrevaya. Towards racially unbiased skin tone estimation via scene disambiguation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2022. 48

[14] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2, 1901. doi: 10.1080/14786440109462720. 6

[15] Baris Gecer, Alexandros Lattas, Stylianos Ploumpis, Jiankang Deng, Athanasios Papaioannou, Stylianos Moschoglou, and Stefanos Zafeiriou. Synthesizing coupled 3d face modalities by trunk-branch generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020. 22, 24

[16] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Luthi, Sandro Schönborn, and Thomas Vetter. Morphable face models-an open framework. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018. 22

[17] Yuliya Gitlina, Giuseppe Claudio Guarnera, Daljit Singh Dhillon, Jan Hansen, Alexander Lattas, Dinesh Pai, and Abhijeet Ghosh. Practical measurement and reconstruction of spectral skin reflectance. In *Computer Graphics Forum*. Wiley Online Library, 2020. 25

[18] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2nd edition, 2002. 43

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014. 2, 9, 13, 22

[20] P Gotardo, J Riviere, D Bradley, et al. *Practical Dynamic Facial Appearance Modeling and Acquisition*. ACM SIGGRAPH Asia, 2018. 25

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 18, 20, 45

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 9

[23] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 15

[24] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 23

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image trans-

lation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 16, 35

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 36, 37, 38, 52, 78

[27] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 1988. 27

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. viii, 14, 16, 23, 54

[29] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 14

[30] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. viii, 14, 16, 23, 33, 34, 76, 78

[31] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko

Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. viii, 14, 16, 23

[32] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 9

[33] Alexandros Lattas, Stylianos Moschoglou, Stylianos Ploumpis, Baris Gecer, Abhijeet Ghosh, and Stefanos Zafeiriou. Avatarme++: Facial shape and brdf inference with photorealistic rendering-aware gans. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 3

[34] Alexandros Lattas, Stylianos Moschoglou, Stylianos Ploumpis, Baris Gecer, Jiankang Deng, and Stefanos Zafeiriou. Fitme: Deep photorealistic 3d morphable model avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 3

[35] Ruilong Li, Karl Bladin, Yajie Zhao, Chinmay Chinara, Owen Ingraham, Pengda Xiang, Xinglei Ren, Pratusha Prasad, Bipin Kishore, Jun Xing, and Hao Li. Learning formation of physically-based face attributes. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 22, 23, 24, 33, 39, 40, 41, 42, 60

[36] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 2017. URL `https://doi.org/10.1145/3130800.3130813`. 7, 22

[37] Gao Lin, Liu Feng-Lin, Chen Shu-Yu, Jiang Kaiwen, Li Chunpeng, Yukun Lai, and Fu Hongbo. Sketchfacenerf: Sketch-based facial generation and editing in neural radiance fields. *ACM Transactions on Graphics*, 2023. 23

[38] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Proceedings of International Conference on Computer Vision (ICCV)*, 2018. 23

[39] Huiwen Luo, Koki Nagano, Han-Wei Kung, Qingguo Xu, Zejian Wang, Lingyu Wei, Liwen Hu, and Hao Li. Normalized avatar synthesis using stylegan and perceptual refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 42

[40] Michele Merler, Nalini Ratha, Rogerio S Feris, and John R Smith. Diversity in faces. *arXiv preprint arXiv:1901.10436*, 2019. 48

[41] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, pages 3481–3490. PMLR, 2018. 34

[42] Christian Murphy, Sudhir Mudur, Daniel Holden, Marc-André Carbonneau, Donya Ghafourzadeh, and Andre Beauchamp. Appearance controlled face texture generation for video game characters. In *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, MIG '20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381710. doi: 10.1145/3424636.3426898. URL https://doi.org/10.1145/3424636.3426898. 24

[43] Christian Murphy, Sudhir Mudur, Daniel Holden, Marc-André Carbonneau, Donya Ghafourzadeh, and Andre Beauchamp. Artist guided generation of video game production quality face textures. *Computers & Graphics*, 2021. doi: https://doi.org/10.1016/j.cag.2021.06.004. URL `https://www.sciencedirect.com/science/article/pii/S0097849321001199`. 24

[44] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 23

[45] Gabriella Pangelinan, Xavier Merino, Samuel Langborgh, Kushal Vangara, Joyce Annan, Audison Beaubrun, Troy Weekes, and Michael C King. The chroma-fit dataset: Characterizing human ranges of melanin for increased tone-awareness. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024. 25

[46] Foivos Paraperas Papantoniou, Alexandros Lattas, Stylianos Moschoglou, and Stefanos Zafeiriou. Relightify: Relightable 3d faces from a single image via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 3

[47] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 35, 36, 37, 38, 52, 78

[48] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas 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*, 2009. doi: 10.1109/AVSS.2009.58. 5

[49] Aashish Rai, Hiresh Gupta, Ayush Pandey, Francisco Vicente Carrasco, Shingo Jason Takagi, Amaury Aubel, Daeil Kim, Aayush Prakash, and Fernando De la Torre. Towards realistic generative 3d face models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024. x, 2, 22, 24, 53, 58

[50] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3D faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2, 23, 31

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)*. Springer, 2015. 17

[52] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. doi: 10.1007/s11263-015-0816-y. 21

[53] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016. 19, 20

[54] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 23

[55] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 23

[56] Irwin Sobel and Gary Feldman. A 3×3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*, 1973. 28

[57] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 45

[58] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 23

[59] William Thong, Przemyslaw Joniak, and Alice Xiang. Beyond skin tone: A multidimensional measure of apparent skin color. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 23

[60] Norimichi Tsumura, Hideaki Haneishi, and Yoichi Miyake. Independent-component analysis of skin color image. *JOSA A*, 1999. 25

[61] Norimichi Tsumura, Nobutoshi Ojima, Kayoko Sato, Mitsuhiro Shiraishi, Hideto Shimizu, Hirohide Nabeshima, Syuuichi Akazaki, Kimihiko Hori, and Yoichi Miyake. Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin. In *ACM SIGGRAPH 2003 Papers*. 2003. 25

[62] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3, 24

[63] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 40

[64] S Yamaguchi, S Saito, et al. *High-fidelity Facial Reflectance and Geometry Inference from an Unconstrained Image*. ACM Transactions on Graphics, 2018. 39

[65] Shuai Yang, Liming Jiang, Ziwei Liu, and Chen Change Loy. Styleganex: Stylegan-based manipulation beyond cropped aligned faces. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023. 23

[66] Xu Yao, Alasdair Newson, Yann Gousseau, and Pierre Hellier. A latent transformer for disentangled face editing in images and videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 23

[67] Longwen Zhang, Qiwei Qiu, Hongyang Lin, Qixuan Zhang, Cheng Shi, Wei Yang,

Ye Shi, Sibei Yang, Lan Xu, and Jingyi Yu. Dreamface: Progressive generation of animatable 3d faces under text guidance. *ACM Transactions on Graphics*, 42, 2023. doi: 10.1145/3592094. 3, 24

[68] Gaspard Zoss, Prashanth Chandran, Eftychios Sifakis, Markus Gross, Paulo Gotardo, and Derek Bradley. Production-ready face re-aging for visual effects. *ACM Transactions on Graphics (TOG)*, 2022. 23

Appendix

A.1 Baseline Implementation

In Section 4.1.1, we detailed the implementation of the Baseline model. The initial model, largely influenced by StyleGAN2 [30], utilized a singular generator to produce outputs across seven channels—three for the skin colour map $\mathcal{M}$, one for the high-frequency detail map $\mathcal{H}$, and three for the position map $\mathcal{P}$. To manipulate the physical characteristics of the model subjects, including gender, ethnicity, and age, we employed the scheme described below:

- Each class was represented as a class vector representation (l_g, l_e, l_a) e.g., $male \rightarrow l_g = [0, 1]$ $female \rightarrow l_g = [1, 0]$. The same goes for other classes as well in terms of ethnicity and age.
- We then multiply the class vectors in weighted vectors (W_g, W_e, W_a), which will result in a latent class encoding e_g, e_e, e_a.
- The latent encodings e_g, e_e, e_a are then concatenated as a single latent encoding and then passed to the generator as a conditioning vector.

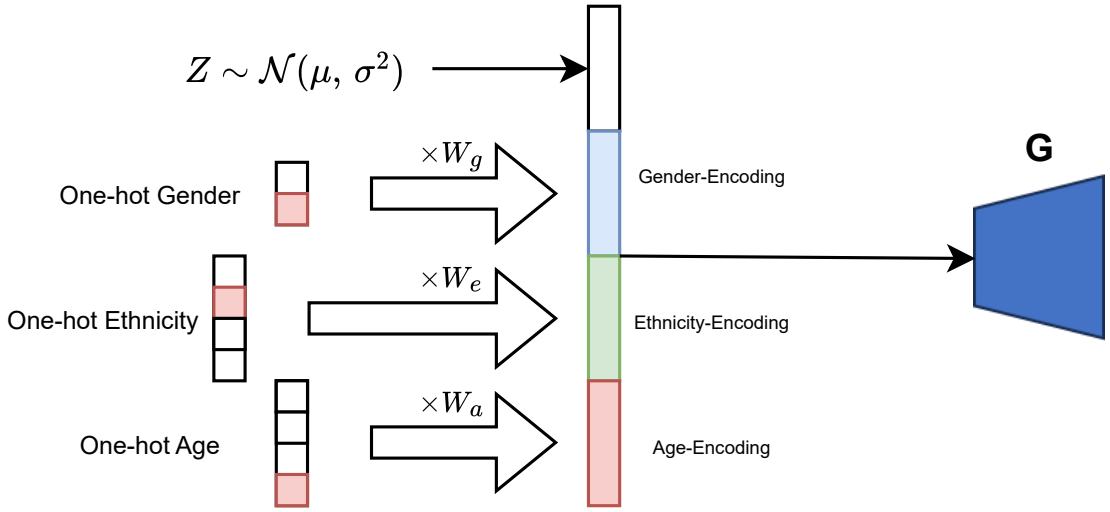


Figure A.1: Conditioning is done by encoding the one-hot representation to a class encoding, and by concatenating the encoding and passing them as the conditioning vector, we can't determine the class.

- The class vector representations are not forced to one-hot vectors; for example, if we want to smoothly transition from a male subject to a female subject, we can play with l_g. For example, $l_g = [0.5, 0.5]$ is for a subject where the gender is something between the two.

A.2 Baseline-Laplacian Implementation

We used the same approach as in A.1 for conditioning the generated face avatar on a specific class. The main difference in the implementation, as mentioned in 4.1.2, is that it uses the Laplacian of the position map as an additional signal for improving the quality of the generated assets and boosting the convergence speed.

A.3 Implementation Details

- Geometry Generator $\mathcal{F}$: We used the same implementation techniques mentioned in the original paper [2].

- Texture generator G: We followed the same training scheme in StyleGAN2 [30], with the hyper-parameter $\gamma = 4$, with the additional change of using three discriminators instead of one, which all three discriminators have an equal coefficient in the final loss function. The model is trained for about 1500 epochs, with a batch size of 8 and a learning rate of 0.002. The input data was also normalized to have a mean of 0 and a variance of 1.

- Skin tone control $G_{\mathcal{A}}$: An image-to-image translation network [26] was used, in addition to the original implementation inspired from [47], we used the multi-patch, multi-resolution discriminator was the model was trained for 300 epochs with the batch size of 4. The initial learning rate was 0.0001, and we used the decay scheduling of 0.1 for each 60 epoch. The input data was also normalized to have a mean of 0 and a variance of 1. $G_{\mathcal{A}}$ was trained of 256×256 feature images.

- Fine-grained detail editing $G_{\mathcal{C}}$: Similar to $G_{\mathcal{A}}$ an image-to-image translation network [26] was used, in addition to the original implementation inspired from [47], we used the multi-patch, multi-resolution discriminator was the model was trained for 300 epochs with the batch size of 2. The inputs (smooth skin colour map $\mathcal{A}$ and high-frequency map $\mathcal{H}$) were upsampled using a bi-linear interpolation from 256×256 to 512×512, and for the output we used a higher resolution reflectance map $\mathcal{C}$ (there was no need to upsample since we already had it in our dataset). We

noticed using a higher resolution created more fine-detailed results compared to its lower-resolution counterpart. The input data was also normalized to have a mean of 0 and a variance of 1.

- Specular Map estimator G_{ϵ_s}: We used the exact same implementation used in [10].
- Normal Map estimator G_{ϵ_n}: Similar to G_{ϵ_s} we used the same implementation in [10].