Appearance Controlled Face Texture Generation for Video Game Characters

Christian Murphy
Concordia University
Montreal, Canada
christiansmurphy@gmail.com

Sudhir Mudur
Concordia University
Montreal, Canada
mudur@encs.concordia.ca

Daniel Holden
Ubisoft La Forge
Montreal, Canada
daniel.holden@ubisoft.com

Marc-André Carbonneau
Ubisoft La Forge
Montreal, Canada
marc-andre.carbonneau2@ubisoft.com

Donya Ghafourzadeh
École de technologie supérieure
Montreal, Canada
gh.dony@gmail.com

Andre Beauchamp
Ubisoft La Forge
Montreal, Canada
andre.beauchamp@ubisoft.com

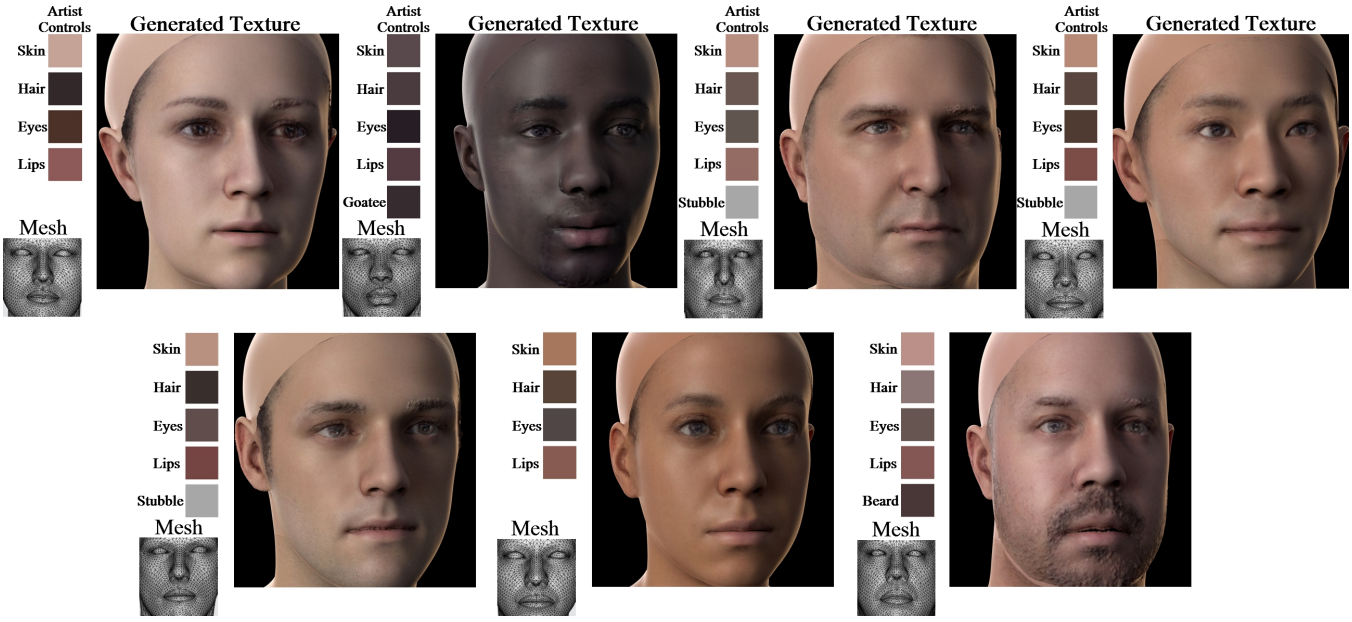

Figure 1: Examples of realistic 4096×4096 **resolution face textures and displacement maps generated from the chosen color values, face mesh, age, and gender.**

ABSTRACT

Manually creating realistic, digital human heads is a difficult and time-consuming task for artists. While 3D scanners and photogrammetry allow for quick and automatic reconstruction of heads, finding an actor who fits specific character appearance descriptions can be difficult. Moreover, modern open-world videogames feature several thousands of characters that cannot realistically all be cast and scanned. Therefore, researchers are investigating generative models to create heads fitting a specific character appearance description. While current methods are able to generate believable head shapes quite well, generating a corresponding high-resolution and high-quality texture which respects the character's appearance description is not possible using current state of the art methods.

This work presents a method that generates synthetic face textures under the following constraints: (i) there is no reference photograph to build the texture, (ii) game artists control the generative process by providing precise appearance attributes, the face shape, and the character's age and gender, and (iii) the texture must be of adequately high resolution and look believable when applied to the given face shape. Our method builds upon earlier deep learning approaches addressing similar problems. We propose several key additions to these methods to be able to use them in our context, specifically for artist control and small training data. In spite

MIG '20, October 16–18, 2020, Virtual Event, SC, USA

ACM ISBN 978-1-4503-8171-0/20/10…$15.00
https://doi.org/10.1145/3424636.3426898

of training with a limited amount of training data, just over 100 samples, our model produces realistic textures which comply to a diverse range of skin, hair, lip and iris colors specified through our intuitive description format and augmentation thereof.

CCS CONCEPTS

• **Computing methodologies → Texturing**.

KEYWORDS

face texture generation, artist controlled character creation, image-to-image translation, fine facial features

ACM Reference Format:
Christian Murphy, Sudhir Mudur, Daniel Holden, Marc-André Carbonneau, Donya Ghafourzadeh, and Andre Beauchamp. 2020. Appearance Controlled Face Texture Generation for Video Game Characters. In *Motion, Interaction and Games (MIG '20), October 16–18, 2020, Virtual Event, SC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3424636.3426898

1 INTRODUCTION

Modern video games often contain character generation systems which the player can use to create their own avatar and for designers to generate additional non-player-controlled characters. However, key characters may require unique appearances which are too complex or detailed to be created using these systems. Instead, the appearance of these important characters is either created by hand, requiring a vast amount of work, or a real life actor who matches the appearance must be found and consent obtained for them to be 3D scanned - a task that can often be difficult and expensive.

With games evolving to include more and more characters with distinctive roles and larger crowd sizes, new ways of generating characters are being explored to cut down on the amount of effort currently required by the creation process. Using multiples of the same artist created characters in crowds or for various roles is a simple way to scale up game size without increasing artist workload, however, immersion breaks can happen for the player if they detect a "clone" [Oxspring et al. 2013]. McDonnel et al. [2009] show that using different face textures and face geometry are two of the most effective techniques for disguising clones, but creating these variations are very time consuming when using traditional methods. Generative techniques for facial geometry have matured considerably [Egger et al. 2019] while the generation of high quality face textures has proven to be much more difficult. Therefore, the application of generative deep learning techniques to face texture generation has been receiving a lot of attention in recent years.

While current techniques focus either on unconditioned random generation from noise or generation of specific faces from photographic input, our goal is to generate realistic and high resolution textures guided by artists' choices. These include age and gender, various appearance attributes, and a face shape generated by the method of Ghafourzadeh et al. [2020]. The face shape condition is important to include so that the network can generate a texture and displacement map which will look natural once applied to the face shape.

Our artists designed the appearance description to include: skin color, hair color, iris color, lip color, and facial hair color and style to generate the texture and displacement map from. As there is no photograph reference to base the texture upon, our problem is highly under-constrained and so the generator needs to precisely and accurately create a large amount of detail not provided by the input. Achieving this level of detail is particularly difficult with small training datasets such as the one we have available for this work. It consists of just 126 samples compiled from scans used in recent game productions. Our dataset has already taken years to create from scanning consenting individuals of different ages, ethnic backgrounds, and appearances, so enlarging it by a large factor is not feasible.

Our solution starts by framing the problem as an image-to-image translation problem since this has been shown to have very good results with small datasets [Isola et al. 2017]. Image-to-image translation is based off the presumption that the location of the pixels in the input and output images are highly correlated, so we modify our data to take advantage of this paradigm. Our meshes and descriptions are transformed into a structure which matches our textures and displacement maps by using a predefined UV layout, allowing us to use this data for image-to-image translation. We then modify the Pix2Pix architecture [Isola et al. 2017], a popular image-to-image translation method, to include recent work in the face texture generation field. Finally, we conduct several experiments to validate our method and present our results.

Our main contributions are:

(1) Modification of the Pix2Pix image-to-image translation method by adding symmetric and global consistency to help generate more realistic textures and displacement maps.
(2) A novel method to transform artistic conditions into an appearance attribute map which artists can easily use to control the appearance of generated textures.
(3) A simple data augmentation strategy to strengthen the network's understanding of the connection between appearance description inputs and texture outputs, enabling the generation of diverse face textures even with a small and biased dataset.

2 RELATED WORKS

Our method builds upon recent deep-learning solutions created in the image-to-image translation, photograph-to-texture completion, face texture and face image generation fields. Below, we briefly review earlier work on these topics.

2.1 Image-to-Image Translation

Isola et al. [2017] generate realistic, synthetic images by inputting semantic labeled images to a U-Net architecture [Ronneberger et al. 2015] with dropout added. The dropout adds noise to the upsampling layers, thus making the network stochastic. Their generator is trained by combining $L1$ pixel-wise loss and discriminator loss so that realistic outputs can be generated even for small datasets. Their PatchGAN discriminator is trained on input-output pairs so that it can also learn the meanings of the semantic labels. Their combined network architecture and training method is commonly referred to as Pix2Pix.

Wang et al. [2018] modify Pix2Pix to be better suited for high resolution images, leading to the name "Pix2PixHD". They add a

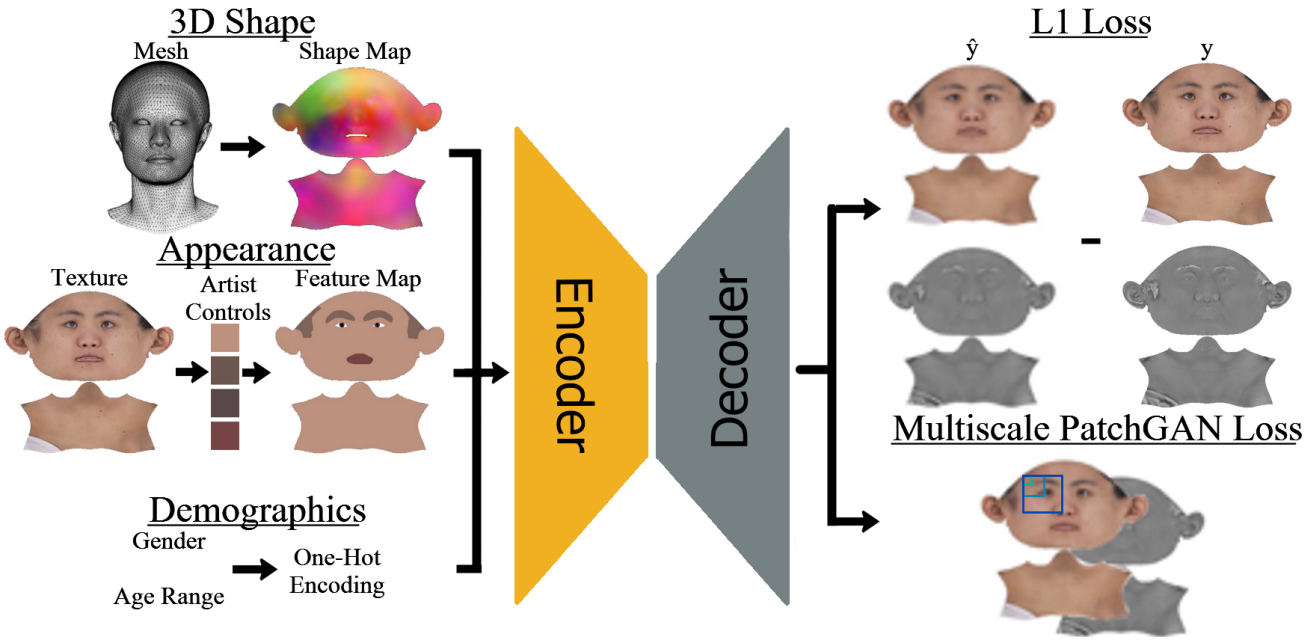

Figure 2: Our solution transfers 3D shape and basic appearance attributes into the same UV format as the target textures and displacement maps so that image-to-image translation can be used. An automated process simulates the artist controls by transforming ground truth textures into a basic appearance representation to use as network input. Our network is trained with pixel-wise $L1$ loss and discriminator loss of multiple different sized patches which we show with early training results.

second generator which learns to upsample the output of the generator used in [Isola et al. 2017] and also add two more discriminators which work on different subsampled versions of the output. The extra discriminators help increase global consistency and sharpen details.

Park et al. [2019] create learned modulation parameters in the normalization layers to preserve semantic information, which lets them achieve good quality results without requiring an encoder in their architecture. They show large improvements over Pix2PixHD [Wang et al. 2018] on large datasets with widely varying image content, however, improvements are modest in problems with smaller and more focused data sets (e.g. Cityscapes [Cordts et al. 2016]).

2.2 Photograph-to-Texture Completion

Occlusion is one of the main problems when transferring photographs to face textures. Therefore, much work has been focused on how to best fill in their unknown regions. Saito et al. [2017] complete textures by analyzing the known parts of the texture and finding its relation to textures in a database to infer what the rest of the texture should be. A similar method is done by Dessein et al. [2015] where they fill in occluded regions with similar known textures and then perform Poisson blending of the edges. Using image completion methods not specific to face textures, like the work by Iizuka et al. [2017], has also been suggested.

Qu et al. [2019] use face hallucination on textures to fill in the occluded regions. The GANFIT method by Gecer et al. [2019b] deals with occlusion by using the photograph as a guide for a face texture generator instead of filling in a texture after the transfer process. Deng et al. [2018] first use the U-Net [Ronneberger et al. 2015] architecture for identity preserved face texture completion by using the loss from a pre-trained classification network.

Yamaguchi et al. [2018] also use the U-Net architecture [Ronneberger et al. 2015] for texture completion, but they introduce a modification to the network which they call "Feature Flipping", where the mirror reflection of the output tensor at each layer is appended to itself to give their textures better symmetry. We add this modification to the Pix2Pix network since it accomplishes the same goal of identity preservation as [Deng et al. 2018] without needing a pre-trained network.

2.3 Face Texture Generation

The 3DMM method by Blanz and Vetter [1999] is a seminal work in the face mesh and texture generation fields through the use of PCA. Booth et al. [2018] use an updated version of the 3DMM method on 10,000 face scans, yet their generated textures still have the same issues as they did years prior: although the textures look natural for the mesh, they are far too blurry to use for modern applications. Sucontphunt et al. [2010] create more realistic textures by searching their texture database using descriptive user inputs, like a sketch, and then performing a linear combination of the closest matches. However, performing linear combinations of images destroys a lot of important micro-details in the skin and hair, again resulting in a smooth texture.

Slossberg et al. [2018] are the first to confront this problem by using a GAN to synthesize a texture, and then use the texture to estimate a face model for it using the 3DMM method. Gecer et al.

[2019a] increase the quality of deep learning texture generation by using a "Trunk-Branch" modification of the Progressive Growing of GANs method by Karras et al. [2017]. The modification allows them to create a texture, normal map, and face shape at the same time so that each of the maps has detailed, corresponding features. Li et al. [2020] improved on previous methods by allowing anatomical and physical attributes along with gender and age to be used as conditions for a mesh and texture generator, but these conditions do not include appearance attributes required by artists such as skin or hair color.

Some issues may arise while creating a 3D face dataset, and if not dealt with, the incorrect data may find its way into generated textures through the discriminator learning that real textures have these issues. Shamai et al. [2019] work to fix this issue by simply masking the loss from known corrupted regions, however this still requires someone to search through the database to find and mask off all of the corrupted data.

2.4 Face Image Generation

Generative models designed for face image generation use network architectures and training methods which could be applied to face texture generation as well. Proof of this is shown in UV-GAN [Deng et al. 2018] which uses a discrimination method originating in the work by Li et al. [2017] for filling unknown regions of face images.

Artists could use written descriptions for generating face textures by re-purposing the FTGAN by Chen et al. [2019]. The LinesToFacePhoto method by Li et al. [2019] is able to create realistic faces fitting a sketch description by adding self-attention to the Pix2Pix architecture. This method could provide a way for artists to input conditions to a generator network, but the time it would take for artists to create the sketches by hand is long and it would not be useful for generating textures with specific color values.

StyleGAN [Karras et al. 2019] is able to generate very realistic face images from two input image controls. Generated face images could then be used in the facial reconstruction method Sela et al. [2017] to create a high quality synthetic mesh and texture. However, this still raises concerns over consent from the people in the control images since we require consent from everyone we scan even though their data is modified afterwards to preserve anonymity.

3 METHODOLOGY

3.1 Overview

The main goal of our work is to create a system which game artists can use to generate high quality face textures which have the appearance and fine details they desire, while also looking natural when applied to a specified face mesh. The output for our deep-learning generator network is therefore a texture and displacement map, and the input conditions for the network are an artist's description of the character's appearance, a face mesh, and the gender and age of the character. The steps we take to be able to use these conditions within the image-to-image translation framework and the training of our network is presented in Figure 2. In the following sections we first describe our model, followed by our training procedure, our input and output parameterization, and our augmentation method.

3.2 Model Architecture

We base our model on Pix2Pix [Isola et al. 2017], which is a U-Net architecture [Ronneberger et al. 2015] with 50% dropout in some decoder layers to introduce noise to the network. We modify the network for face texture generation by including Feature Flipping at each layer as in [Yamaguchi et al. 2018]. This is implemented in our network by creating a Mirror Block with a forward operation which returns a concatenation of the input and its mirrored reflection. The Mirror Blocks are placed after each layer in the network except the final. In the decoder, the Mirror Blocks are placed after the dropout operations to keep information loss symmetrical. Each individual modification to Pix2Pix [Isola et al. 2017] and our complete network architecture can be viewed in Figure 3.

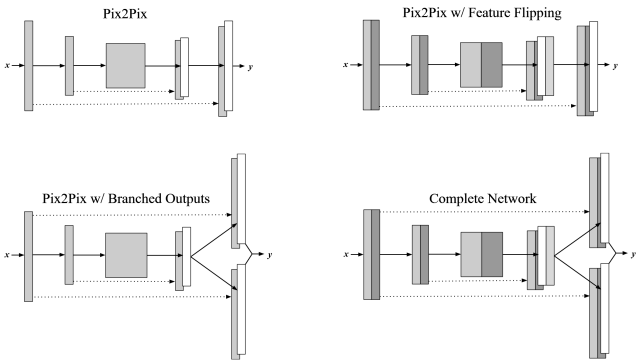

Figure 3: Our network modifies Pix2Pix [Isola et al. 2017] by adding the mirror reflection of tensors, called Feature Flipping, at each layer and branched output layers for the texture and displacement map.

We also add a branched output layer as in [Gecer et al. 2019a] to provide separate output layers for the texture and displacement map. Gecer et al. [2019b] propose that this helps the model better handle multiple types of outputs while still keeping them coherent with each other. We initialize a Branch Block by specifying the amount of 2D transpose convolutional layers to be created, which is two in our case. The Branch Block feeds the same tensor to each of the layers individually, and the output of each is concatenated and returned.

Three PatchGAN discriminators trained on different resolutions of the outputs are used as described by Wang et al. [2018]. We include three discriminators rather than one because face textures should have globally coherent and well defined details. The second and third discriminator process patches of 50% and 25% bilinearly subsampled versions of the data respectively.

3.3 Training

The generator G maps x {shape, appearance, age, gender} and dropout noise z to produce $\hat{y}$ {texture,displacement map}. $L1$ loss between $\hat{y}$ and the target output y is used along with the Binary Cross Entropy loss of three PatchGAN discriminators, D_1, D_2, D_3, to train G. Our complete training objective with multiscale discriminators

and $L1$ loss on the conditional generator G is defined as:

$$G^* = (\min_G \max_{D_1, D_2, D_3} \sum_{k=1,2,3} \mathcal{L}_{cGAN}(G, D_k)) + \lambda \mathcal{L}_{L1}(G) \quad (1)$$

Where $L1$ loss on a conditional GAN G is defined as:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1] \quad (2)$$

And the loss from a discriminator D is:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z))] \quad (3)$$

With an adversarial min-max objective of:

$$G^* = \min_G \max_D \mathcal{L}_{cGAN}(G, D) \quad (4)$$

Each network uses normal weight initialization [Glorot and Bengio 2010] and all are trained using the Adam optimizer [Kingma and Ba 2014] with a learning rate of 0.0002 and momentum parameters β_1 at 0.5 and β_2 at 0.999. The $L1$ loss weight λ is set to 100. The loss used to train the discriminators is halved to train these networks slower than the generator, as is done in Pix2Pix [Isola et al. 2017]. The networks are trained for 100K steps with a batch size of 8 using an Nvidia RTX 2080TI, taking roughly 30 hours. The textures had imperceptible changes when trained for any longer. At inference time, a texture and displacement map is produced in 5 milliseconds using the GPU.

3.4 Feature Map

To give our artists control over the appearance, we need the input of the network to have more specific information given to it rather than simple one-hot encoded attributes such as "blonde hair" or "green eyes". Exact color values for each attribute could be given as a simple vector, but the proper use of these values would likely be difficult to learn without any context. Therefore, we encode the appearance information in a meaningful format in the form of a UV map. This allows the network to quickly and easily understand the spatial relationship between appearance conditions through the use of image-to-image translation. To be able to do this, we first created a Segmentation Map which defines where certain features appear in the textures and then fill these regions with the appearance information of each sample. These segmented regions include: hair, eyebrows, iris, lips, and facial hair locations (above lips, chin, lower cheeks and neck).

We created our Segmentation Map by tracing over a face texture which included hair at all of the possible facial hair locations, and then expanded the eyebrow region so that it would cover the location of the eyebrows for any texture. This had to be done because the eyebrows had the most spatial variance compared to the other segmented regions since its location depends on the sample's brow bone structure.

Color values were then taken directly from the textures to be used as the desired attribute color for that region in the Segmentation Map. For example, a sample's skin color is chosen by simply using the value of the pixel in the middle of the forehead on the texture. This chosen skin color is then used to replace all of the pixel values in the skin region of the Segmentation Map. Similarly, we identified a specific location in each segmented region so that the process could be done automatically for our entire dataset. We also show how this process works in Figure 4.

Before creating the Feature Maps, we went through the dataset and marked down the facial hair style of each sample so that we could use this information during the creation process. Based on the facial hair style, we know whether or not to use the colors chosen to represent the beard, moustache, and goatee regions while creating the Feature Map. In the case of a sample having stubble, we fill in all facial hair regions with a chosen grey color.

Since the generator is not globally aware of its appearance conditions but still has to place some skin in hair filled regions, we also include the desired skin color in hair-filled regions. This is done by using the skin color instead of the hair color for every fourth pixel when filling in these regions. However, if the participant has stubble, then the hair and skin color is alternated after every pixel so that the generator would learn the difference between a grey beard condition and a grey stubble condition.

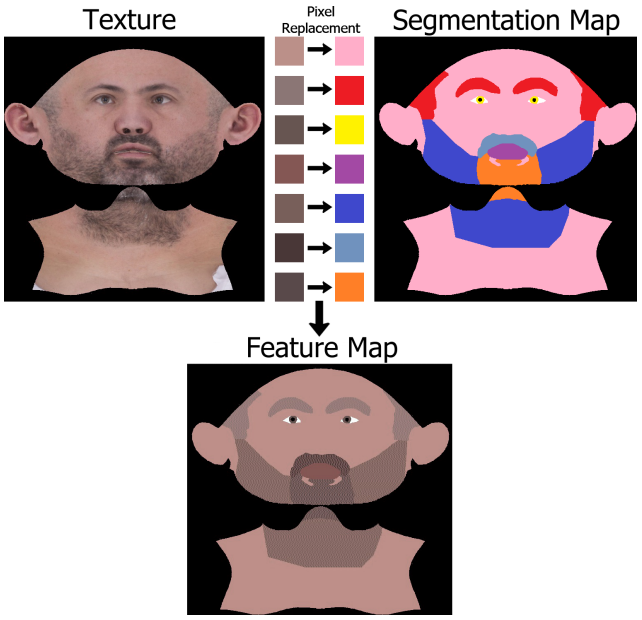

Figure 4: Pixels from the texture are automatically selected to override the pixels in each region of the Segmentation Map, resulting in a Feature Map.

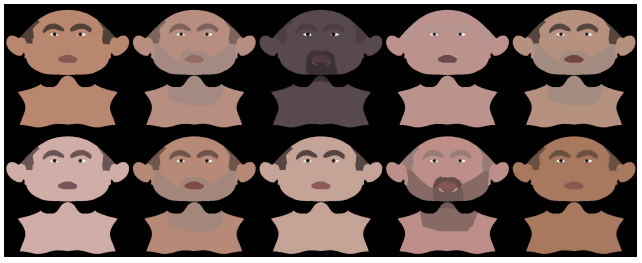

Figure 5: The Feature Maps created from the textures of our test set.

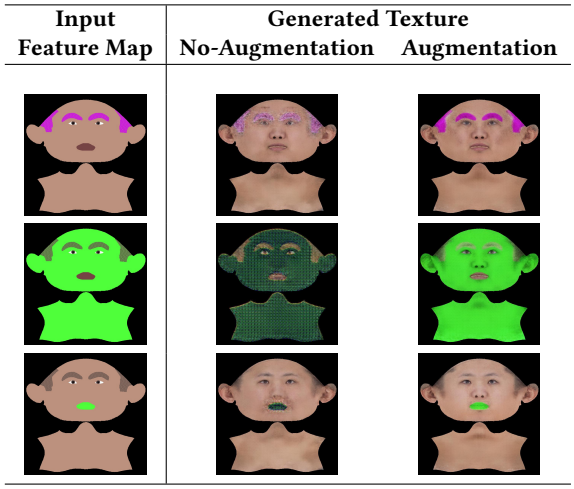

Figure 6: Regions of a real Feature Map were replaced by an unnatural value and given to two identical networks, but one is trained with augmentation and the other without. Our augmentation method makes the network generate a texture based on any given value.

This system allows artists to embed their desired appearance attributes into a Feature Map quickly and easily since they only have to specify a minimal amount of colors, only one for each region, and a facial hair style. The Feature Map is then made near instantly through our algorithm, allowing for it to be input to the generator quickly so that the user receives visual feedback from their choices in realtime.

3.5 Data Augmentation

When training with a limited dataset, networks tend to quickly memorize the expected output for each input rather than learning to generalize for the problem. We augment our data during training to prevent memorization so that the network can learn the conditions better by introducing a 15% chance to apply data augmentation. Three scalars between 0 and 1 are randomly sampled from a uniform distribution, one for each channel in the images. Multiplying the channels in the Feature Map and texture by the same scalars changes the images in the same way since the Feature Map was created from the values in the texture.

Even this simple data augmentation method greatly helps avoid over-fitting on such a small dataset. The effect that augmentation has on the network is presented in Figure 6 by comparing the textures generated from synthetic inputs in the same network trained with and without augmentation. Instead of producing textures with noise in the locations where unnatural color values not attainable through augmentation are present, these colors are actually used when generating the textures, showing that the network understands the Feature Map condition better.

3.6 SuperResolution

We use a trained SuperResolution network as a post-processing step on the generated texture and displacement map once the artist has chosen their final appearance description values. The network performs a $\times 4$ supersampling of the initial 1024×1024 resolution images to have a finalized 4096×4096 resolution texture and displacement map ready for production use. Many recent works such as [Li et al. 2020; Wang et al. 2018] achieve high resolution outputs by training a SuperResolution network at the same time as the generator. However, we train the SuperResolution network separately to be able to use techniques which would be unavailable to us if it was trained at the same time as the generator network. This includes the ability to train the network using small, randomly chosen patches of data and to include data from larger, public datasets of similar domains like face images.

We train the MSRResNet architecture used as the baseline model for the AIM 2019 SuperResolution Challenge [2019] following the training methodology by Ledig et al. [2017] to supersample 64×64 resolution patches of our textures/displacement map pairs and face images from the Chicago Face Database [Ma et al. 2015] to 256×256 resolution. This face image dataset was used since its data acquisition method was very similar to ours, being that all of the face images were taken with a standardized camera position, lighting

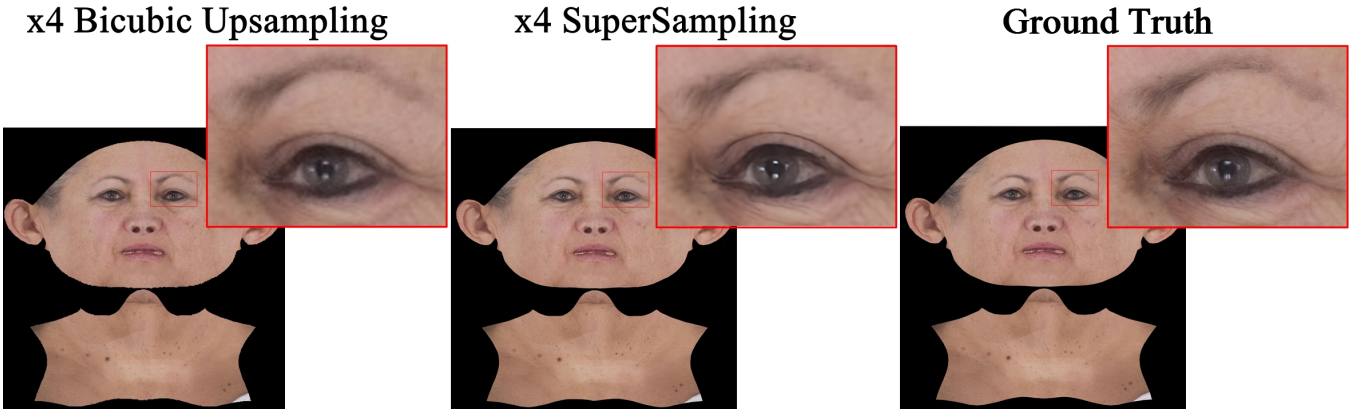

Figure 7: Our trained SuperResolution network performs very well in comparison to bicubic upsampling for enhancing the facial features of this held-out texture, especially on the wrinkles, hair, freckles, and eyes.

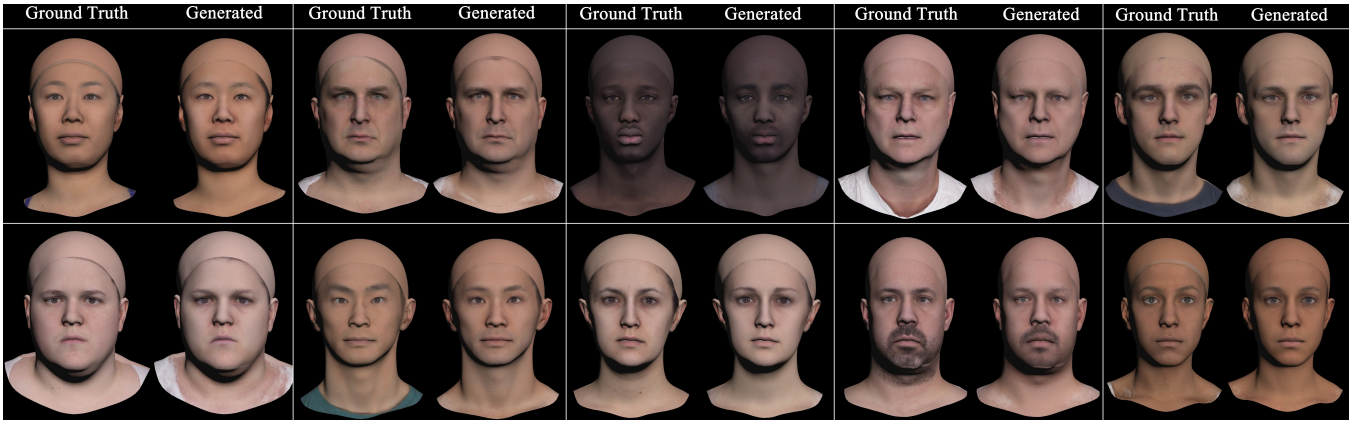

Figure 8: From just a few selected colors, personal attributes, and a provided mesh, our generator is able to synthesize new textures and displacement maps which have a high resemblance to our diverse held-out test set, suggesting that our network generates realistic and accurate appearances for unseen data.

setup, subject position and facial expression. Patches were chosen randomly from ground truth data, billinearly downsampled to 25% resolution and had a 50% chance to apply either horizontal or vertical flipping. We alternate the use of texture/displacement map pairs and face images after each training step and use two PatchGAN discriminators by Isola et al. [2017], where one discriminates the texture/displacement map pair patches while the other discriminates the face image patches. The performance of our model can be viewed in Figure 7.

Table 1: The demographics of our dataset.

Ethnicity	#
Caucasian	78
East Asian	17
Black	10
Hispanic	13
South Asian	8

Age	#
20-29	37
30-39	34
40-49	23
50-59	12
60-69	13
70+	7

Gender	#
Female	62
Male	64

4 DATASET

For 126 people, we record the demographics about each person and reconstruct a high quality mesh, texture, and displacement map using our own proprietary photogrammetry technology. We use terms described by Raj Bhopal [2004] to describe the sample's ethnicity, and show our dataset's ethnicity, age range, and gender figures in Table 1. We do not use the ethnicity as a condition for our network and instead only record the ethnicity to understand the bias in our dataset, which, as shown in Table 1, is heavily biased toward Caucasian samples.

Our meshes, textures, and displacement maps all have the same topology and UV format created using the R3DS Wrap commercial solution for simplifying head reconstructions. Our test set was then created through a random stratified sampling from each ethnic group to create a test set with 10 samples which includes at least one sample from each ethnicity and nearly every age range. These samples were held out during training. Figure 8 shows the ground truth test set textures and displacement maps applied to their corresponding mesh. The Feature Maps created from these test set textures can be viewed in Figure 5.

4.1 Shape Map

We represent our mesh data as a Shape Map by following the method described by Gecer et al. [2019a] to encode the vertex positions in image space. Using a Shape Map instead of the mesh directly makes the 3D shape be interpreted better by convolutional neural networks. The Shape Map is made by normalizing the complete set of aligned meshes and then transferring their vertex values into the UV map format. Triangle interiors are filled in using barycentric coordinates so that each pixel represents a point in 3D space. This process is shown in Figure 9.

5 EVALUATION

In this section we present an evaluation of our method including a presentation of our test set results, an ablation study of our proposed methods, and an analysis of the trained model.

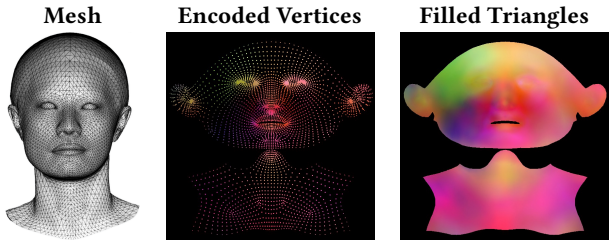

Figure 9: Shape Maps are made by encoding the 3D position of each vertex in the mesh into image space using its UV format and then filling the triangle interiors.

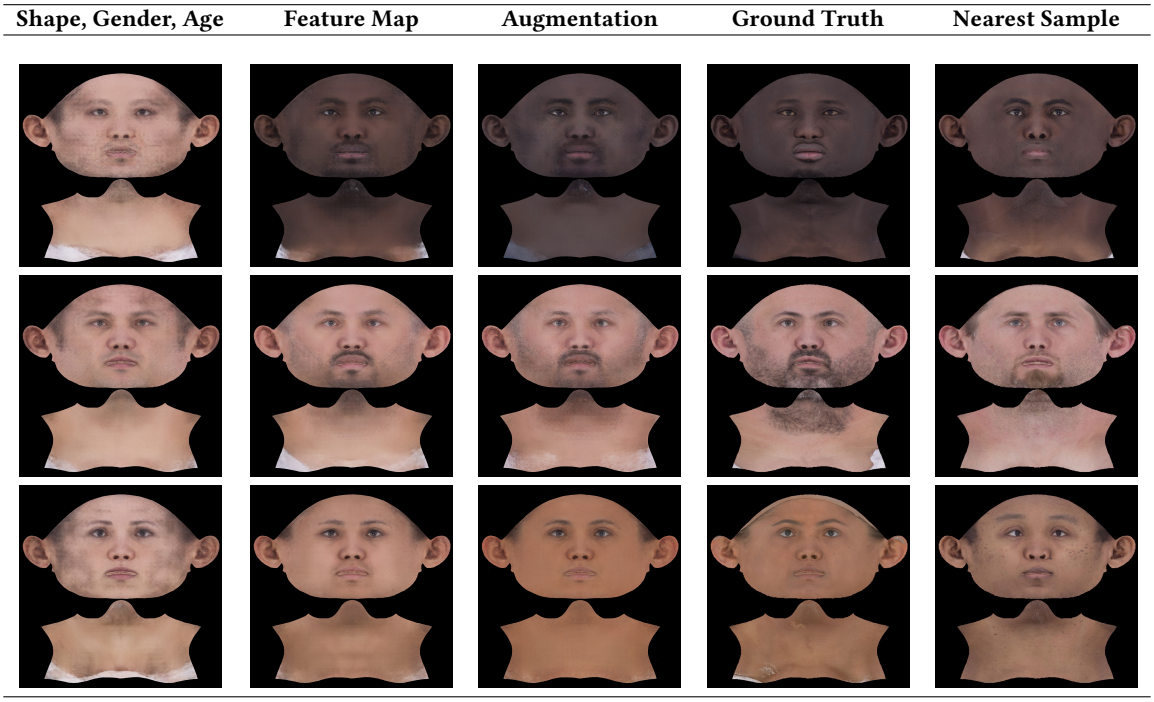

Figure 10: Generated textures from each step in our ablation study. Textures lack consistency without the use of a Feature Map, and unseen input values are better utilized after training with augmentation. Synthesized textures have appearances far from the closest texture in the training set based on RMSE.

5.1 Results

The descriptive conditions used by our generative model allows for textures with appearances similar to ground truth to be produced. By comparing the appearance of the given meshes with the ground truth and generated textures and displacement applied to them, shown in Figure 8, we see that the generated textures have a very similar appearance to ground truth because of the skin, eye, lip, and hair colors included in the Feature Map. Facial hair details like stubble enhances the level of similarity even more, suggesting that artists will have enough freedom in input appearance options to make textures very close to the appearance they desire. We also provide a more detailed look at the quality of results we obtain in Figure 1.

5.2 Ablation of Methods

We show how three of our sample's texture predictions change during an ablation study of our methods in Figure 10. The study begins by training our network using the Shape Map, age, and gender conditions to generate the texture and displacement map. This is similar to other works [Gecer et al. 2019a; Li et al. 2020] which primarily correlate face shape to texture appearance. However, given our requirement of artist control, this is not a viable method to synthesize textures with a desired appearance. Based on these results, we also believe that using only these conditions for generating textures would require a much larger dataset than ours to obtain quality results, which as explained earlier is very time consuming to create.

We retrained our network with the Feature Map included, shown in column 2. This new condition addresses both of the aforementioned problems very well. It helps make a much more cohesive textures similar to the ground truth appearances in column 4, but they do not exactly match the given Feature Map values. We trained our network once more to include our data augmentation method, shown in column 3. Using this augmentation method helped the network create textures with skin, hair, lip, and eye colors which nearly matches those in the held out test set. In the last column of Figure 10, we include the training set sample with the lowest RMSE distance to the final generated texture to show that our trained model is synthesizing unique textures, not simply modifying a sample from the training data.

5.3 Trained Model Analysis

To visualize the performance of our final fully trained model, we perform a sensitivity analysis on the Feature Map and Shape Map conditions. This is done by taking the inputs of a sample, changing one, and analyzing the resulting change. We modify the input to the network by interpolating between one of a sample's inputs, x_0, and the same type of input from another sample, x_1, to create a new input $\hat{x} = \alpha x_1 + (1 - \alpha)x_0$, where α is the interpolation factor. Using interpolation to assess the model also gives the benefit of simulating the actions an artist might be doing when choosing Feature Map values and modifying their given mesh.

Feature Map interpolation results are presented in Figure 11 and Shape Map interpolation results are in Figure 12. Note that the

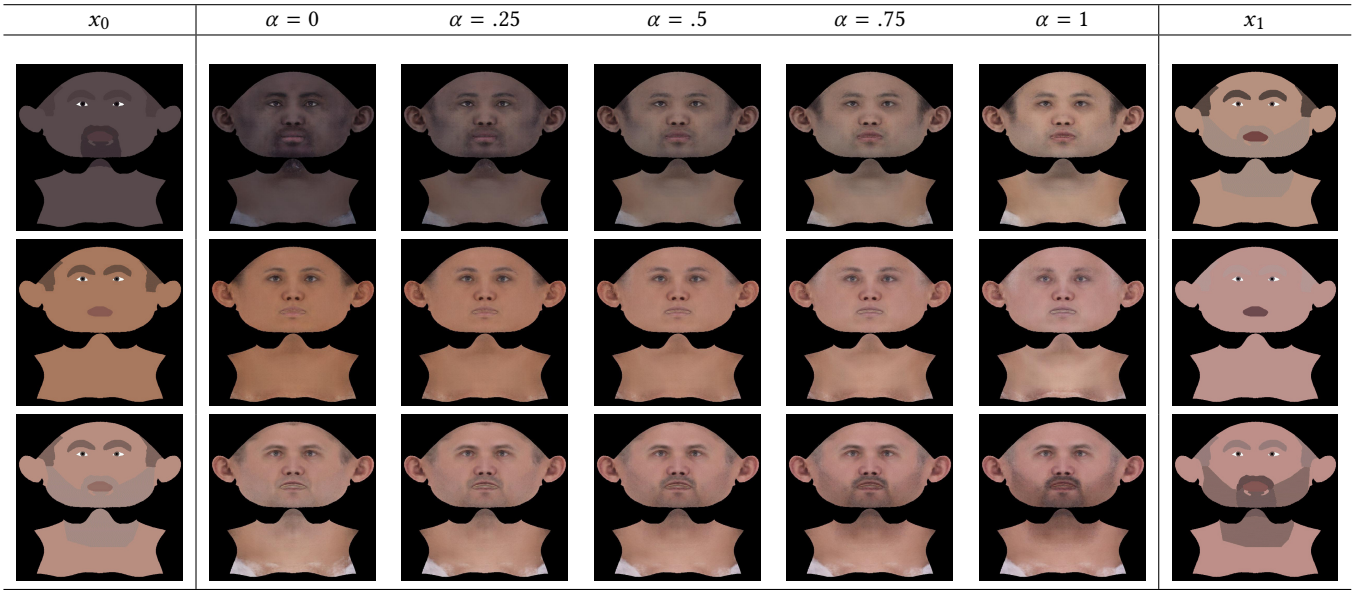

Figure 11: Results from the linear interpolation between two Feature maps, x_0 and x_1, by the value α. The color values should be changing to match the Feature Map, while the features which depend on shape should be static.

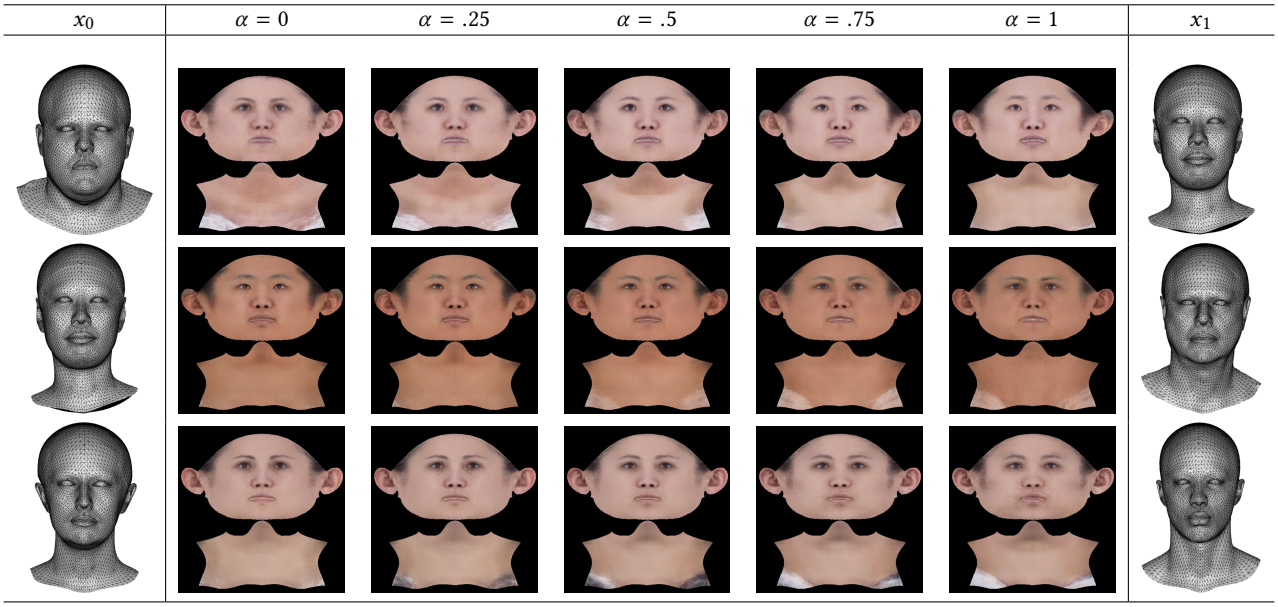

Figure 12: Results from the linear interpolation between the Shape Maps of two different meshes, x_0 and x_1, by the value α. The shape of facial features should change to fit the mesh, but the color of them should be static.

mesh is presented instead of the Shape Map for better visualization of the differences. Interpolating between Feature Maps shows that the model uses the condition very well and can generate realistic textures from input combinations not in the training set. Generated textures are affected very differently by each input type, showing that the network has learned a disentangled representation of facial feature properties. The network has learned how to combine features which depend on shape with those which depend on the Feature Map in a natural way, as all of the generated textures are realistic looking.

Finally, we generated a few meshes with anthropometric measurements outside of our face mesh dataset range using the Part-based 3DMM by Ghafourzadeh et al. [2020]. The shape maps from these meshes were input to the trained network along with a variety of artist created feature maps and the chosen gender and age.

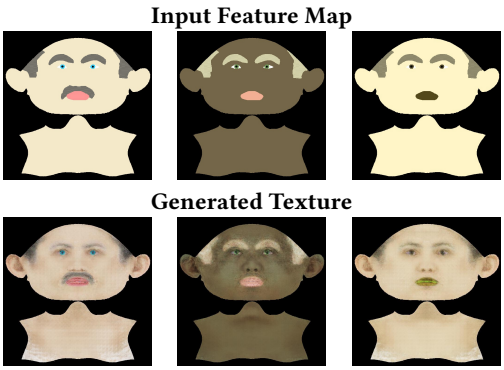

Figure 13: Some out-of-distribution Shape and Feature Map combinations can lead to textures being generated with checkerboard artifacts and a green tint in skin and lip color.

The results were then ×4 SuperSampled and applied to the original mesh, with the resulting renders shown in Figure 14. These generated textures still met all of the input conditions with a high level of detail even though the given meshes were far different than those trained with.

6 LIMITATIONS AND FUTURE WORK

The main limitation we see in the proposed method is in the lack of facial details like freckles, acne, and wrinkles being generated in the textures. This likely comes from the combination of not directly requesting these details in the network input, having discriminators which are not picking up on the importance of having these details for discrimination, and from the network lacking the understanding of how these minute details are related to face shape, age, and gender.

Detecting these skin details and adding them onto both the textures and Feature Maps of another sample using image editing methods like [Clément et al. 2007] could be a way to have the network train with these details clearly specified in the input. To have the network better understand how age relates to these skin details, our data could be passed through the face aging simulation by Suo et al. [2007] on our younger samples and adding the results to our dataset with their new simulated age. To learn how gender relates to skin details, CycleGAN [Zhu et al. 2017] could be trained to modify a texture to look like it is from the opposite gender so that the results could be used to augment our dataset. Researching such dataset augmentation methods would be novel for the face generation field and could be a promising way for current solutions to improve their results.

We also show some failure cases in Figure 13 where realistic Shape and Feature Map inputs were provided. In one case, the resulting textures had checkerboard artifacts. Also shown is a dark brown color given in the Feature Map resulting in a green color tone on the cheeks and lips. The cheeks and lips are often brighter than the given skin color, so the green color is likely a result of brightening the brown color given.

It would also be interesting to apply our segmentation based Feature Map generation method to different 3D models used in games, such as full body textures and clothing textures. This would be a first step towards the complex problem of generating a character's complete 3D model and texture from a simple artist description.

7 CONCLUSION

In this work, we addressed the problem of generating completely new, high resolution, natural looking face textures for game characters given their appearance attributes, face shape, age and gender. From the ablation studies and sensitivity analysis performed on our proposed method, we find that our system is a simple and reliable solution for generating natural looking 4096 × 4096 resolution face textures and displacement maps from artist inputs - even when having a relatively small training dataset available to us.

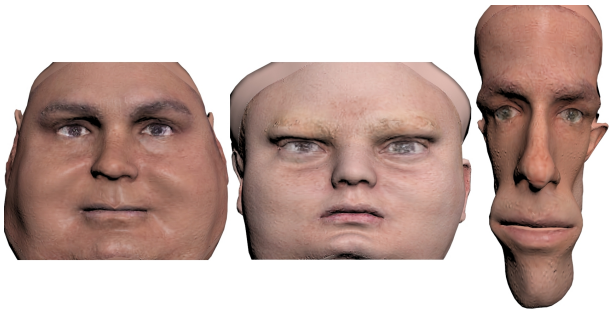

Figure 14: Our network still creates pleasant results even when given meshes generated far outside of the dataset distribution.

REFERENCES

Raj Bhopal. 2004. Glossary of terms relating to ethnicity and race: for reflection and debate. *Journal of Epidemiology & Community Health* 58, 6 (2004), 441–445.

Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 187–194.

James Booth, Anastasios Roussos, Allan Ponniah, David Dunaway, and Stefanos Zafeiriou. 2018. Large scale 3D morphable models. *International Journal of Computer Vision* 126, 2-4 (2018), 233–254.

Xiang Chen, Lingbo Qing, Xiaohai He, Xiaodong Luo, and Yining Xu. 2019. FTGAN: A Fully-trained Generative Adversarial Networks for Text to Face Generation. *arXiv preprint arXiv:1904.05729* (2019).

Olivier Clément, Jocelyn Benoit, and Eric Paquette. 2007. Efficient editing of aged object textures. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. 151–158.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3213–3223.

Jiankang Deng, Shiyang Cheng, Niannan Xue, Yuxiang Zhou, and Stefanos Zafeiriou. 2018. Uv-gan: Adversarial facial uv map completion for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7093–7102.

Arnaud Dessein, William AP Smith, Richard C Wilson, and Edwin R Hancock. 2015. Example-based modeling of facial texture from deficient data. In *Proceedings of the IEEE International Conference on Computer Vision*. 3898–3906.

Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 2019. 3D Morphable Face Models–Past, Present and Future. *arXiv preprint arXiv:1909.01815* (2019).

Baris Gecer, Alexander Lattas, Stylianos Ploumpis, Jiankang Deng, Athanasios Papaioannou, Stylianos Moschoglou, and Stefanos Zafeiriou. 2019a. Synthesizing Coupled 3D Face Modalities by Trunk-Branch Generative Adversarial Networks. *arXiv preprint arXiv:1909.02215* (2019).

Baris Gecer, Stylianos Ploumpis, Irene Kotsia, and Stefanos Zafeiriou. 2019b. Ganfit: Generative adversarial network fitting for high fidelity 3d face reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1155–1164.

Donya Ghafourzadeh, Cyrus Rahgoshay, Sahel Fallahdoust, Andre Beauchamp, Adeline Aubame, Tiberiu Popa, and Eric Paquette. 2020. Part-Based 3D Face Morphable Model with Anthropometric Local Control. In *Graphics Interface*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–14.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4401–4410.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

Ruilong Li, Karl Bladin, Yajie Zhao, Chinmay Chinara, Owen Ingraham, Pengda Xiang, Xinglei Ren, Pratusha Prasad, Bipin Kishore, Jun Xing, et al. 2020. Learning Formation of Physically-Based Face Attributes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3410–3419.

Yuhang Li, Xuejin Chen, Feng Wu, and Zheng-Jun Zha. 2019. LinesToFacePhoto: Face Photo Generation From Lines With Conditional Self-Attention Generative Adversarial Networks. In *Proceedings of the 27th ACM International Conference on Multimedia*. 2323–2331.

Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. 2017. Generative face completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3911–3919.

Debbie S Ma, Joshua Correll, and Bernd Wittenbrink. 2015. The Chicago face database: A free stimulus set of faces and norming data. *Behavior research methods* 47, 4 (2015), 1122–1135.

Rachel McDonnell, Michéal Larkin, Benjamín Hernández, Isaac Rudomin, and Carol O'Sullivan. 2009. Eye-catching crowds: saliency based selective variation. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–10.

Sean Oxspring, Ben Kirman, and Oliver Szymanezyk. 2013. Attack on the clones: managing player perceptions of visual variety and believability in video game crowds. In *International Conference on Advances in Computer Entertainment Technology*. Springer, 356–367.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2337–2346.

Chengchao Qu, Eduardo Monari, Tobias Schuchert, and Jürgen Beyerer. 2019. Patch-based facial texture super-resolution by fitting 3D face models. *Machine Vision and Applications* 30, 4 (2019), 557–586.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. 2017. Photorealistic facial texture inference using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5144–5153.

Matan Sela, Elad Richardson, and Ron Kimmel. 2017. Unrestricted facial geometry reconstruction using image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*. 1576–1585.

Gil Shamai, Ron Slossberg, and Ron Kimmel. 2019. Synthesizing facial photometries and corresponding geometries using generative adversarial networks. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 15, 3s (2019), 1–24.

Ron Slossberg, Gil Shamai, and Ron Kimmel. 2018. High quality facial surface and texture synthesis via generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 0–0.

Tanasai Sucontphunt, Borom Tunwattanapong, Zhigang Deng, and Ulrich Neumann. 2010. Crafting 3d faces using free form portrait sketching and plausible texture inference. In *Proceedings of Graphics Interface 2010*. 209–216.

Jinli Suo, Feng Min, Songchun Zhu, Shiguang Shan, and Xilin Chen. 2007. A multi-resolution dynamic model for face aging simulation. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8798–8807.

Shugo Yamaguchi, Shunsuke Saito, Koki Nagano, Yajie Zhao, Weikai Chen, Kyle Olszewski, Shigeo Morishima, and Hao Li. 2018. High-fidelity facial reflectance and geometry inference from an unconstrained image. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Kai Zhang, Shuhang Gu, Radu Timofte, Zheng Hui, Xiumei Wang, Xinbo Gao, Dongliang Xiong, Shuai Liu, Ruipeng Gang, Nan Nan, et al. 2019. Aim 2019 challenge on constrained super-resolution: Methods and results. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 3565–3574.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.