UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2022

Title: *Conditional Generative Adversarial Networks for 3D Face Generation with Expression Control*

Author: Luc van den Handel

Project Abbreviation: 3D-GAN

Supervisor(s): Prof. Deshen Moodley

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	5
Experiment Design and Execution	*0*	*20*	15
System Development and Implementation	*0*	*20*	10
Results, Findings and Conclusions	*10*	*20*	15
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	**80**

Conditional Generative Adversarial Networks for 3D Face Generation with Expression Control

Luc van den Handel
University of Cape Town
Cape Town, South Africa
vhnluc001@myuct.ac.za

ABSTRACT

A large amount of research has gone into the generation of unique faces using Generative Adversarial Networks (GANs). However, most of this research has been in the generation and manipulation of images of faces with 3D face generation remaining as a fairly unexplored problem domain. To this end the 3D Expression Conditional Generative Adversarial Network (3DEC-GAN) was investigated. This GAN was designed to be simple, avoiding the need for internal models and mesh operations, and to work directly on point cloud data. Post evaluation it was discovered that 3DEC-GAN is able to generate faces in general, with promising results from a quantitative analysis, but still lacks the ability to generate finer details needed for the faces to be usable. It was further discovered that despite this failure, 3DEC-GAN was successfully conditioned on facial expressions, displaying an ability to reproduce the expressions of: neutral, bare teeth, cheeks in and mouth open.

CCS CONCEPTS

• **Generative Adversarial Network** → **3D Face Point Cloud Generation**; • **PointCloud** → *Surface Reconstruction*; • **GAN evaluation metrics** → FID,KID,Generalisation,Specificity; • **GAN** → training stability.

KEYWORDS

generative adversarial networks, face generation, point clouds, evaluation of 3D objects

1 INTRODUCTION

The human face is central to human life and our society, with the ability to recognise and process faces having its own, dedicated, neuron pathways in the brain[12]. As such the face is an extremely efficient mechanism for conveying information about a person, from their age and gender to their current emotions. Faces are also present in most forms of media and entertainment from video games to movies, where a large amount of work is usually done to ensure that these faces look correct, both from a purely behavioural and proportional perspective - avoiding the uncanny valley[7] - and that they conform to the creative team's definition of what that person should look like and what emotions they are experiencing. This process normally involves teams of artists and graphics experts and is the source of a large cost to productions, in both time and money. To this end the generation of high definition 3D faces with control over their general features qualifies for investigation.

The generation of unique samples is a problem domain that has garnered fair deal of research, with many successes being seen through the use of machine learning techniques. Critically Generative Adversarial Networks (GANs) [8] are highly applicable, especially considering their innate creativity and usability as, typically, all that is required to generate a sample is noise (a vector, z, with the correct shape and populated with random values). Therefore, they were considered for the problem domain of face generation.

Subsequently, GANs have seen success in the problem domain of face generation. However, most of the applications have involved the generation and manipulation of images of faces, with little attention on 3D faces. The research that has been done on 3D faces also involves highly complex networks which make use of a number of internal structures and models - such as autoencoders and morphable models [4, 20]. Therefore this project aims to investigate the design of a much simpler GAN that works directly on the 3D data in the form of point clouds. Ideally this GAN should be able to generate 3D faces that are both directly usable in real world applications and are statistically correct (free from errors such as mode collapse) with the incorporation of feature control.

2 BACKGROUND

2.1 Generative Adversarial Networks.

In the field of machine learning, there are a vast number of algorithms and systems which each learn to perform a specific task. Generative Adversarial Networks (GANs) are one of these algorithms [8]. They belong to the neural network class of algorithms and are originally unsupervised learners - meaning that they do not require labelled data to learn, just real examples. They function by training two separate and opposing neural networks, the discriminator and the generator. The discriminator has the function of classifying samples as either real or fake and the generator has the function of generating said fake samples. The generator then trains to fool the discriminator while the discriminator trains to be more accurate in its knowledge of the difference between fake and real samples. As such these two networks play a learning game, where they are constantly trying to outmatch each other, until convergence is reached. At convergence the generator is creating samples that are indistinguishable from the real samples in the sense that the discriminator has exhausted its ability to gain new knowledge of the difference between real and fake samples and therefore has a 50% probability of being correct. This is because the probability distributions of the generator and the real data have become extremely similar (with a minimal distance between them).

The GAN loss function, which encodes the training game played between the discriminator and generator, can be seen in (1) below:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))] \quad (1)$$

Once convergence has been reached training can end and the resulting generator can be used to create output samples which are unique but mostly indistinguishable from the real data - the sample may have never been seen before, but could seamlessly belong to the training data set.

There is a large range of possible outputs from a GAN, however one of the most common applications is in image generation and manipulation, with outputs ranging in complexity from handwritten digits[19] to faces[13].

2.2 Deep Convolutional GAN

An early extension to the GAN was the incorporation of convolutional layers into the design of the generator and the discriminator as seen in [19]. Here, Radford et al. used 2D transpose convolutional layers in the generator to build up the output from the input noise, and subsequently in the discriminator, the more common 2D convolutional layers were used to better understand the image and determine if it was fake or not, as seen in other image classification models.

The properties of convolutional layers allow for two key advantages: the use of channels and local understanding. Firstly, while the convolution works in two dimensions on the image data, it provides the possibility of channels - adding an extra dimension to the data - which in the case of [19] allows for the move from grey-scale data to RGB data, where there must be 3 channels for the red, green and blue colour values. Secondly convolution works by considering a subset of pixels at a time in a sliding window over the input space, thus allowing for the understanding of simple constructs and how they are combined to form the final output - for example, in the case of handwritten digits, the network can understand the lines, loops and crosses used to from the digits.

2.3 Face Generation

While GANs and their variants have been used to generate a variety of output data from image processing and generation to speech and music generation [1], a popular application has been the generation of images of faces, or so-called 2D face generation [13]. This problem domain has seen great success with the creation of models able to generate high resolution samples, with accurate facial details and proportions and the ability to exert control over the generated face in that details such as age, race, gender, etc. can be specified for the generated face.

This displays that GANs are capable of understanding the structure and detail of a face in 2 dimensions and in 3 channels. However, the real world applications of 2D face generation are limited. Therefore, it has called into question the possibility of 3D face generation, which has many real world applications from game design, to CGI, to virtual reality. It is this line of questioning which inspired this project.

2.4 Conditional GANs

The Conditional Generative Adversarial Network (cGAN) [15] is another natural extension of the original GAN which allows for a degree of control over the output of the generator. This is achieved via the use of a conditional information vector which encodes label data and is appended to the network inputs. While this control enhances the usefulness of the GAN, it does cause a departure from unsupervised learning as a cGAN must be provided with labelled data to train and is therefore classified as a supervised learning algorithm.

The conditioning is achieved via the use of classes, where each class defines a general subset of the full range of possible outputs from a particular GAN. For example, a GAN trained on the MNIST handwritten digits data set[5, 8, 15] can make use of a class for each digit, 1 through 9. Here, each class also corresponds to a label in the training data.

It stands to reason that conditioning may also be applied to a 3D face generation with the goal of achieving expression and/or identity manipulation. Where the vector determines general aspects of the face and the noise determines the finer facial details that make the face unique.

If we consider the conditional information vectors to be y and y' for the discriminator and generator respectively, the GAN loss function is updated as seen in equation (2) below:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x+y)] + \mathbb{E}_{z\sim p_z(z)}[1-\log D(G(z+y'))] \tag{2}$$

3 RELATED WORK

Through the following review of existing literature, it is observed that Machine Learning has been applied to 3D face generation. However it typically does not involve direct operations on point clouds, opting rather to either with 3D Meshes involving the use of complex mesh operations[20] or to learn a morphable model which limits usability and is equally complex[4]. Furthermore, while forms of control have been applied to these generation methods, none have used standard conditioning using labels.

3.1 3D Morphable Models

A 3D Morphable Model (3DMM) is different kind of model that can be used for 3D face generation and is based on the concept of Principle Component Analysis [3]. 3DMMs operate by forming a standard model that is a statistically average, or general, face which can then be manipulated by specifying the values of a range of input parameters. Here instead of generating a whole new face, this standard face is adjusted (morphed) according to the parameters.

While 3DMMs on their own are powerful tools in the field of 3D face generation, their parameters are often difficult to understand, their range of outputs can be limited and they lack the inherent creativity of a GAN. As such they have seen wide success as internal components of GANs for applications in 2D face generation where an internal 3D understanding is required - such as pose correction or manipulation[26].

3.2 CoMA

Convolutional Mesh Autoencoders (CoMA) [20] are a class of neural network which were designed to generate 3D faces. They differ from the GANs as they learn by iteratively down-scaling a real sample into a specified vector, then up-scaling back up to the original size

and comparing the result of the up-scaling to the original sample. Once training is complete, the up-scaling half can be used with noise as input - in the same way one might use the generator of a GAN - to generate 3D faces. Ranjan et al. applied this architecture to 3D face generation and was thus one of the first examples of success in this problem domain where there was no reliance on a 3D Morphable Model (3DMM). This was achieved via the use of complex mesh operations that were used to down-scale and up scale 3D mesh object.

CoMA has become the basis from which to conduct research into 3D face generation, being a direct point of comparison for both this project and MeshGAN [4]. This is largely due to the availability of both the data used to train CoMA and the accessibility of CoMAs data preprocessing pipeline. Using these a comparable model can be constructed where the input training data and desired outputs are identical to that of CoMA.

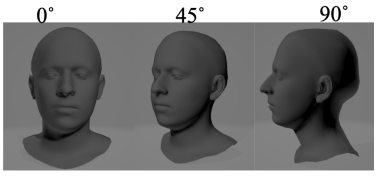


Figure 1: Showing the rendering of a sample from the raw data of a face in the neutral expression (in 3 different views - horizontal camera rotations).

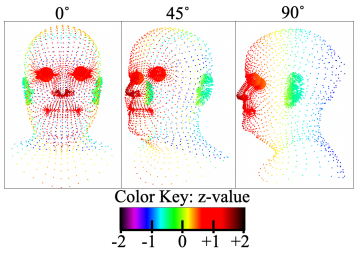


Figure 2: Showing a sample point cloud as a result of the CoMA preprocessing of a face in the neutral expression (in 3 different views - horizontal camera rotations).

3.2.1 Data and Preprocessing. The research team behind CoMA has provided their implementation as open source software with an MIT Licence along with the raw data that was used through a separate licence agreement. As such the facial scans that were used for this project were obtained from CoMAs website once permission was granted and a user account was created. This raw data consists of time-series scans of 12 participants in 12 expressions where for each expression the participant started with a neutral expression, moved their face to make the expression, then returned to neutral. This data is stored as *PLY* files containing the mesh of a 3D object. The CoMA preprocessing is then responsible for sorting the objects as required (such as by expression), down sampling the objects and converting them to point clouds with a fixed number of vertices. Figures 1 and 2 show the raw data and the result of the preprocessing. It is worth noting that the data has perfect spheres in place of the eyes, most likely due to the eyes themselves not being scan-able by the equipment used.

3.3 MeshGAN

MeshGAN [4] is the foremost GAN for 3D face generation and is the reference study for this project. Cheng et al. formed MeshGAN by combining concepts from many disciplines of machine learning. For example, MeshGAN learns a 3DMM[1] via the use of convolutional mesh operations and an auto-encoder as the discriminator, as seen in CoMA[20].

Therefore, the architecture of MeshGAN is highly complex and it does not necessarily belong to any of the 3 general categories of GAN (conditional, progressive and controllable) but instead makes use of ideas from each of the categories, blending them together.

With MeshGAN, Cheng et al. has furthermore outlined a set of metrics used to evaluate generated 3D faces. A critical step in the development of 3D GANs as these metrics blend new approaches specifically designed to work on 3D point clouds with more established GAN metrics. These same metrics were used in the evaluations conducted for this project are explained in detail in section 5.1. The metrics are: Generalisation, Specificity and Fréchet Inception Distance (FID). Generalisation and Specificity evaluate a specific set of generated samples against a set of, true, testing samples, using euclidean distance in space to understand the fundamental differences between the real and generated data. FID is a metric inherited from 2D GANs that generate images, which uses a pre-trained inception network to produce an evaluation of the models performance, again with respect to a specific set of generated and testing images. However to apply FID to 3D, Cheng et al. establish the idea of rasterising the 3D object into a 64x64 2D image, using lambertian shading, which could then be fed to a pre-existing FID calculation framework which uses Inception Networks made for testing images.

4 DESIGN OF 3DEC-GAN

The 3D Expression Conditional Generative Adversarial Network (3DEC-GAN) is the final GAN studied for this project. It is designed to generate 3D faces in the form of point clouds containing 5023 points in 3D space, where the identity of the face is random and unique but the expression of the outputted face can be declared by the user. It is an extension of a Simple DC GAN - which defines and provides the ability to generate faces without any control - where conditional information vectors were incorporated into the network structure and the training data was refined.

4.1 Simple DC GAN

The Simple Deep Convolutional GAN (Simple DC GAN)[2] is a GAN which is designed to work on 3D point cloud data and generate 3D faces with identity and expression. The goal for this GAN was simple: to confirm that the data can be understood and reproduced by a stable neural network, free from mode collapse.

[1] MeshGAN has the goal of functioning like a 3DMM, where a general face is adjusted to form the desired identity and expression, but is not the same as a 3DMM.
[2] Designed by Liam Watson for PCE-GAN[25], as part of his group work contribution to this project

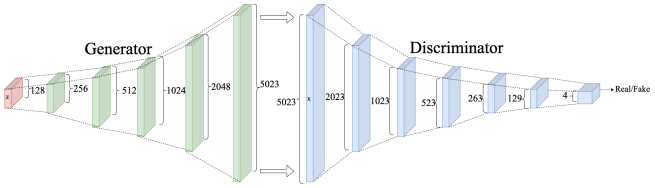


Figure 3: Showing the architecture of the Simple DC GAN, where z is the noise vector and x is face point cloud data. Each layer is a convolutional layer of size nx1 with 3 channels and the LeakyReLU activation function.

The Simple DCGAN is based on the DCGAN produced by Radford et al. in [19] as described in Section 2.2 with a number of fundamental changes to conform to our 3D data format.

Firstly, the differences and similarities between image and point cloud data was considered. A 2D image can be thought of as a 2D array where each element is an array of length 3 with values corresponding to the red, green and blue, color channels. A point cloud is somewhat similar to this where each element consists of three values, except now it's x, y, and z coordinates in 3D space. As such, a point cloud would just be a one dimensional list of these points - or in other words a 1xn "image" where n is the number of vertices in the point cloud. Therefore, the model architecture was modified to use 1D convolutional layers instead of 2D layers but the number of channels remained the same with a value of 3.

Secondly, the sizes of the input and output layers - for the discriminator and generator respectively - had to be adjusted to match the number of points of each face as defined by CoMAs preprocessing pipeline[20]. In our case this was set to be an array of length 5023.

Figure 3 shows the final network design for the generator and discriminator. Many of the network details such as, the size and number of layers in the discriminator and generator and the shape of the input noise were determined through hyper parameter tuning.

4.2 3DEC-GAN

The simple DC GAN was extended to create the final conditional GAN studied: The 3D Expression Conditional Generative Adversarial Network (3DEC-GAN). This extension of the Simple DC GAN involved two major modifications: the incorporation of a conditional information vector into both the discriminator and the generator, and a refinement of the input data.

4.2.1 Conditioning the Faces. In order to correctly make these extensions to Simple DC GAN we need to understand its latent space[3] and what it understands about 3D faces. We can observe that the training data is a collection of 3D faces with a range of identities and expressions. Therefore, theoretically the traversal of the latent space, via the manipulation of the noise vector of the generator, should see changes to the expression and identity of the generated face. As such the conditional information vector needs to function as a mechanism for confining the latent space so that only identity is effected by the value of the noise, sectioning off the latent space into subsets corresponding to the desired expressions.

With this in mind, we can use conditional information vectors concatenated to the inputs of the discriminator and generator - y and y' respectively. This will condition the noise to section off the portion of the latent space where the desired expression can be found. In past work it has been seen that the shape of the conditional information vector matches that of the input being conditioned[15, 18], therefore, y has a length of 5023 and y' has a length of 128, matching the length of the faces and noise respectively. In practice this is achieved via the use of embeddings in each network which, when given a certain label value, produce a vector that will correspond to a label - the values of contained within are a directly correlated to the label value. The vector produced is thus the conditional information vector which can be concatenated to the network inputs. This can be seen in Figures 4 and 5.

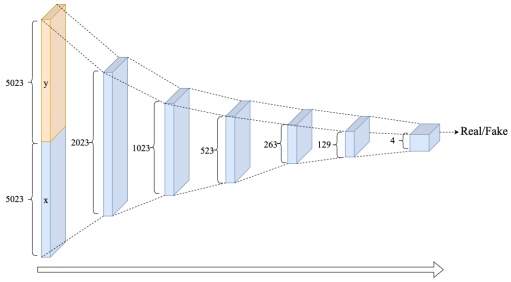


Figure 4: Showing the architecture of the Discriminator of 3DEC-GAN, with conditional information vector y - concatenated onto the input - where each layer is a 1D convolutional layer with 3 channels and length specified.

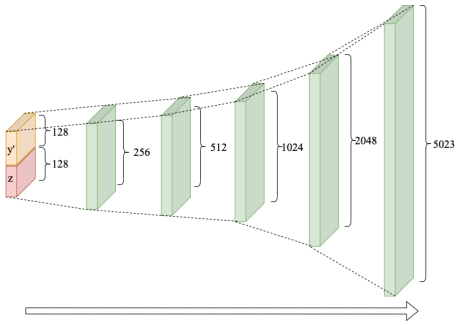


Figure 5: Showing the architecture of the Generator of 3DEC-GAN, with input noise vector z and conditional information vector y', where each subsequent layer is a 1D transpose convolutional layer with 3 channels and length specified.

4.2.2 Training Data. For the embeddings to produce the conditional information vectors, the networks need to be provided with a label value corresponding to the expression of the current face. This can be done by leveraging the directory structure produced by the data preprocessing. Each expression is contained in its own

[3]The latent space is a concept which defines a high-dimensional space in which all the possible outputs of a model exist. A single generated sample is therefore a point in this space, and moving through the space will change the value of the output sample.[23]

directory, therefore, we can load the expressions separately, give each face in these separate lists the correct label, - in our case a number value corresponding to a particular expression - join the lists together and shuffle them. The resulting list of labeled faces can then be used to train 3DEC-GAN.

It is worth noting that the raw data had to be slightly adjusted as each expression for each identity is a time-series as explained in section 3.2.1. Therefore, for the desired expressions the neutral faces needed to be removed and placed in their own directory (allowing for the addition of the "neutral" expression), resulting clean in input data where correct labelling can be guaranteed.

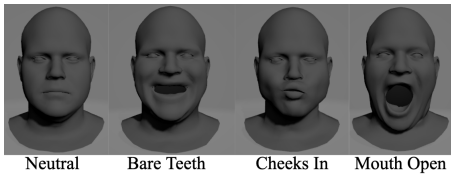


Figure 6: Showing the four chosen expressions from one identity in the raw data.

4.2.3 Expressions. While 3DEC-GAN can theoretically learn any number of expressions - by adding or removing classes - for this project 4 expressions were chosen to be learned: Neutral, Bare teeth, Cheeks In and Mouth Open (in the data this is called "mouth_extreme"). An example of each expression extracted from the raw data can be seen in Figure 6. These expressions were chosen as they are highly distinct from each other with no overlapping facial features (for example, in the original data a "mouth side" expression was included where participants moved their lips as far right and left as possible, but this also involved opening their mouth, a feature that overlaps with the mouth open expression) to minimise the chance of any mode collapses where two or more similar expressions might be merged with some or all expressions being lost.

4.3 Hyperparameters

Table 1: Table showing the hyperparameters used for the final 3DEC-GAN model evaluated

Learning Rate	Batch Size.	No. Epochs
1e-5	4	2000
No. Samples	**No. Expressions**	**Noise Size**
300	4	128
Optimiser	**Criterion**	**Loss Function**
Adam	BCE Loss	Original GAN[8]

Through rapid testing and prototyping, the hyperparameters, as seen in Table 1[4] were determined. These values resulted in a model with the lowest probability of collapse, best guarantee convergence and strongest understanding of each expression. The final model was trained for 2000 epochs. Although, it was observed that after roughly 200-300 epochs the improvement in output quality leveled out, this longer training did provide slight improvements (but was the point where the issue of diminishing returns was encountered).

[4]Note that the same learning rate was used for both the generator and the discriminator and the number of samples are on a per expression basis - i.e. the total number of training samples is samples per expression multiplied by the number of expressions.

4.4 Implementation

The entire system created for this project was implemented using Python[24]. The machine learning library used to create and train the models was PyTorch[17]. This was chosen due to pre-existing familiarity with the library. As previously mentioned, the data preprocessing used was an open source implementation by CoMA. The last notable package used was open3d [27] in the construction of the visualisation framework. Other python packages used are:

(1) Numpy[10] for data management.
(2) Matplotlib[11] to plot data during training
(3) Python Time for thread sleeping
(4) Python Math for metrics calculations
(5) Python os and sys for importing of local object and argument reading.

Lastly, the calcualtions for FID and KID were done using an open source implementation by Abdul Fatir, which can be found here[6], and is based on [21].

5 EVALUATION APPROACH

The approach used for the evaluation of the Simple DC GAN and 3DEC-GAN involves the use of 4 metrics; Generalisation, Specificity, Kernel Inception Distance (KID) and Fréchet Inception Distance (FID), with a secondary visual inspection of the generated samples. An experimental platform is used to streamline both the processed of model training and evaluation along with ensuring consistency between evaluations.

5.1 Metrics

The metrics used for the evaluation of the GANs were chosen due to their use in past work and because they are regarded as the best method for GAN evaluation. Generalisation, Specificity and FID were chosen in order to compare results against MeshGAN [4], and KID was chosen as it is an extension of FID which aims to improve the accuracy of the metric. In order to calculate the metrics, sets of generated and real samples are required, where the bigger the sets, the more accurate the metric. Therefore it was determined that to maximise accuracy while still allowing for the metrics to be calculated in a reasonable amount of time[5] 200 generated samples and a pre-exisiting training set - all the faces of each expression not used in training - would be used.

5.1.1 Generalisation and Specificity. Generalisation and Specificity were calculated based on their description in [4], where per-vertex Euclidean distances were measured[6]. Generalisation is a measure of how well the model understands the general concept of a face based on the training data, as such for each sample in the testing set, its closest match is found in the generated set and the mean

[5]The computational complexity of the algorithms to calculate the metrics is very bad, therefore should the sets be any larger the time required for calculation would be infeasible

[6]As the faces are on a scale of meters, the results were scaled up by a factor of 100 to match the MeshGAN scale of millimeters.

of the per-vertex distances is calculated, the final metric is then the mean and standard deviation of each value obtained for the faces in the testing set, which is called the *Generalisation Error*. The procedure for specificity is very similar, except now the goal is to test the accuracy of specific faces, checking if a generated face is reasonable or can be expected for the given training data, as such, each generated sample has its closest match found. Therefore the *Specificity Error* is the mean and standard deviation of the distance measurements for each face in the set of generated samples.

Essentially, Generalisation and Specificity are enquiries into the latent space of the model via the use of per-vertex euclidean distances. Generalisation is used to measure the overall scope of the space, if it is large enough to contain the general idea of a face and all of its variations. In other words it is a check to see whether the faces that we know should exist in the space are present or not. Specificity is then an a verification of the latent space to check that it hasn't gone beyond the bounds of the desired space, checking if the model has manufactured features that should not exist according to the training data, by evaluating the generated samples against the real data.

5.1.2 FID and KID. FID and KID are both calculated via the use of a pretrained network called an Inception Network[22]. After processing the generated and real samples, distance metrics can then be extracted from the network to determine how close the probability distributions of the generator and the real data are to each other. There are many different kinds of statistical distances that can be reported - hence the use of both KID and FID, which each use their own distance measurement. However, there is an issue with KID and FID in that the inception model has been explicitly trained on images. To account for this the visualisation framework was used to render the faces into 64x64 images of a frontal view of the face in both the generated and real sample sets before KID and FID were calculated. This method was used by Cheng et al. in [4] to calculate FID.

5.2 Visualisation

While the metrics provide evaluations from a mathematical, statistical and theoretical perspective, they can only go so far as to answer the question of whether these GANs can *actually* be used, with regard to the real world applications of 3D face generation. As such the generated samples should be inspected to verify the metrics and assess usability.

Therefore, a visualisation framework is used view generated samples and calculate KID and FID - as mentioned in section 5.1.

The framework provides two main mechanisms for viewing faces: as the raw point cloud and a meshed object. This allows for the data produced by the models to be viewed in an unaltered form while also providing a means for creating objects that are viewed more easily but have gone through some form of post processing

The framework was constructed using Open 3D [27]. The point cloud can be viewed without any processing and using Open 3D's default settings where the color of the point it determined by its z coordinate value (depth), which is its distance from the camera in its initial position - with blue points being far away and red points being close - as seen in Figures 2 and 11. Subsequently, the mesh of the face is obtained by estimating the normals of the point cloud and using Poisson Surface Reconstruction[14] on the points and their normals. The mesh is then rendered using Phong shading and a standard surface color (no texture, instead each polygon is assigned a grey color - RBG: 0.5, 0.5, 0.5)

5.3 Experimental Platform

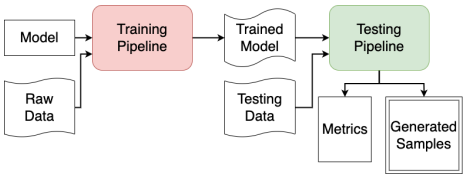


Figure 7: Showing a high-level overview of the experimental platform

The experimental platform[7], as seen in Figure 7, can be broken down into two major components; the training pipeline and the testing pipeline. The training pipeline streamlines the training of new models which conform to the given input and output data format and the testing pipeline is used to produce a standard set of metrics and view generated samples for a given pre-trained model.

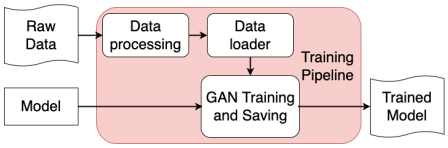


Figure 8: Showing the structure of the training pipeline

5.3.1 Training Pipeline. The training pipeline, as seen in Figure 8, is responsible for all processes surrounding the model's learning; from the processing of the raw data up to the saving of trained models. The pipeline starts by processing the raw data into the desired format as defined by CoMA [20]. The processed data can then be used in the formation of a data loader. The data loader is used to train a given model, where the model weights are saved at regular intervals during training.

For the case of the conditional GAN some adjustments to this standard pipeline had to be made. Such as the incorporation of expression labels in the training data and the refinement of the raw data which was explained in section 4.2.

It is also worth noting that the original training pipeline was implemented independently of the GAN being made, and very early on in the development of the system. Since each GAN has highly unique and specific training requirements, the training pipeline served more as a foundation from which to base a GAN-specific training pipeline.

[7] Completed as a group, where Sebastian Oliver designed and built the training pipeline, Liam Watson designed and built the Simple DC GAN as a baseline from which to build further models and to test the platform early on, and I, Luc van den Handel, designed and built the testing pipeline

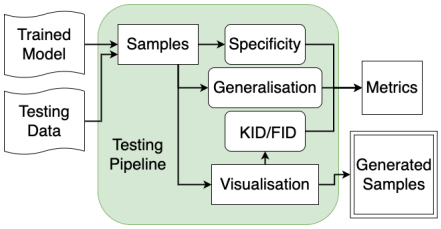


Figure 9: Showing the structure of the testing pipeline

5.3.2 Testing Pipeline. The testing pipeline, as seen in Figure 9, is responsible for everything after model training, this being; experiment data collection and the viewing of generated samples. It does this via the use of our own implementations of specificity and generalisation, a visualisation framework for converting the point clouds to 3D meshes and FID and KID calculations.

The samples used in the calculation of the metrics are a subset of the original data allocated as testing data - so it was not used in training - and a set of generated samples. The generated samples are obtained by loading the learned model weights into a model object and supplying that model with noise to generate faces. For the case of the conditional GAN the pipeline also supplies label information to the model as specified by the user when running the pipeline.

The pipeline also makes use of the visualisation framework for two reasons. Firstly, as motioned in section 5.1.2, the KID and FID calculations require that the face be rasterised into an image, which can be done using the open3d features provided by the visualisation framework. Secondly, as this is the end of the experimental process, it is worthwhile to be able to view the faces produced by the models for a simple, expert, qualitative assessment to be performed by the researcher.

6 EVALUATION

As mentioned in Section 5, the testing pipeline was used for the evaluation of many models, with a focus on the metrics: Generalisation, Specificity, KID and FID. The pipeline also integrates with a visualisation framework that meshes and renders the point cloud to be manually inspected - with the purpose of verifying the metrics and understanding where the model is succeeding and where it is not. The tables and figures below show the metrics calculated and images rasterised by the testing pipeline, first for the simple DC GAN as a verification of its ability to generate faces and secondly of 3DEC-GAN, evaluated on a per-expression basis.

6.1 Simple DC GAN

6.1.1 Metrics. Table 2 shows the quantitative evaluation of the Simple DC GAN which demonstrates the GANs ability to produce 3D faces. It can be seen that on all accounts the model far out performs noise, which is a simulation of a model which produces points in 3D space at random and is theoretically guaranteed to produce metrics that correspond to unacceptable model. These results indicate that the model must be producing point clouds which at the least resemble - or have a tendency to resemble - 3D faces, which gives an indication that the Simple DC GAN may form a strong base from which to create more complex models. Furthermore, it provides insights into what a "good" value for a metric should be, for example, in the worst case the KID value seems to be close to a value of 1.

Table 2: Table showing calculated metrics for samples generated by the Simple DC GAN and noise (theoretically guaranteed poor results).

Model	Gen.(mm)	Spec.(mm)	FID	KID
DC-GAN	0.867±0.175	0.596±0.142	19.79±1.580	0.17±0.007
Noise	161.5±0.002	159.8±1.1	81.37±3.824	1.00±0.010

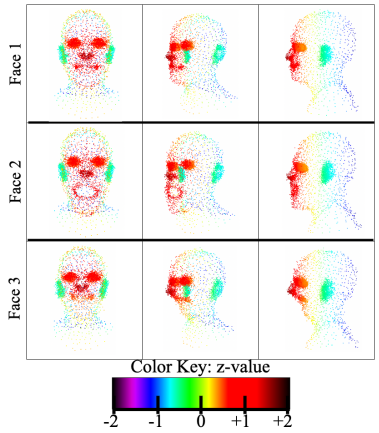


Figure 10: Showing three output samples from the Simple DC GAN, each viewed at the same three angles

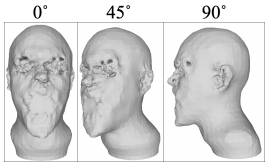


Figure 11: Showing a meshed and rendered sample generated by the Simple DC GAN.

6.1.2 Inspection of Generated Samples. In Figure 10 we can observe the faces generated by the Simple DC GAN. These three faces were specifically chosen as they represent the range of possible faces the GAN is able to produce. They also show that already at this point the GAN is able to understand facial expressions - while its ability to reproduce them correctly is questionable as seen with the clearly incorrect proportions of *Face 3*. These samples confirm, what was suggested by the metrics, that the Simple DC GAN does function as expected and can be used as a reliable base from which to build further GANs for 3D face generation.

Lastly, for completeness, Figure 11 was included showing the output of the Simple DC GAN as a fully meshed 3D object. This face happened to be generated with a *mouth open* expression and is another example of a generated face lacking a degree of correctness with regard to the proportions of the face as the chin is too far forward. It is also displays clear errors in the face as there is clearly a lack of detail and a high degree of noise which will be discussed further in Section 6.2.

6.2 3DEC-GAN

Table 3: Table showing the Generalisation and Specificity results of 3DEC-GAN evaluated per expression type generated.

Expression	Generalization(mm)	Specificity(mm)
Neutral	0.954±0.098	0.849±0.011
Bareteeth	0.879±0.050	0.806±0.010
Cheeks In	1.020±0.088	0.851±0.029
Mouth Open	0.815±0.055	0.777±0.049
Average	**0.917±0.073**	**0.821±0.025**

Table 4: Table showing the KID and FID results of 3DEC-GAN evaluated per expression type generated.

Expression	FID	KID
Neutral	17.69±0.726	0.13±0.002
Bareteeth	19.77±0.519	0.14 ± 0.003
Cheeks In	20.83±0.654	0.17±0.004
Mouth Open	14.60±0.949	0.08±0.002
Average	**18.22±0.712**	**0.13±0.003**

6.2.1 Metrics. Tables 3 and 4 show the calculated metrics for 3DEC-GAN on a per-expression basis, where each value is the mean and standard deviation across the values calculated for each face in the testing set. This was done in order to evaluate how well the model is able to be conditioned and generate specific expressions and to detect if mode collapse has occurred. It has already been shown in Table 2 that the Simple DC GAN - from which 3DEC-GAN is based - is able to generate faces in general. It can be seen that 3DEC-GAN is able to generate all desired expressions, each to a similar degree of accuracy.

These results also show how well 3DEC-GAN is able to produce each expression, as while all the values are similar, the model performs the best when generating the *Mouth Open* expression, and worst when generating the *Cheeks In* expression.

Furthermore, the use of the average of the results allows for a direct comparison to the Simple DC GAN. Here, 3DEC-GAN performs slightly worse on Generalisation and Specificity, but better on KID and FID compared to the Simple DC GAN. This might be explained by the fact that the Simple DC GAN was trained and tested on all identities and expressions in the original data, therefore it is possible that a certain face may both be seen as a correct representation of one expression/identity and a slightly incorrect representation of another. However, an improvement in KID and FID shows that the probability distribution of 3DEC-GAN's generator more closely matches that of the training data, compared to the Simple DC GANs generator. These values are very promising as they can only be a result of the output faces being more accurate from a visual perspective, which is innately human and correlated to the usability of the model.

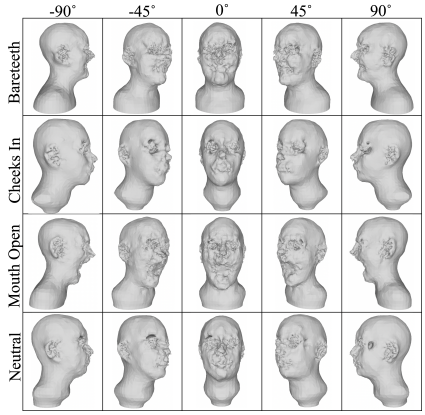


Figure 12: Showing the output from 3DEC-GAN where the noise remained constant while the conditional information vector was varied - i.e. an attempt to generate the same identity in different expressions.

6.2.2 Inspection of Generated Samples. Figure 12[8] displays the quality of the output of 3DEC-GAN. It can be seen that it is able to produce a different face based on the expression condition and that difference is mainly a change in expression and not identity. The faces also posses general features which are correct, accurate and in proportion such as the general shape of head including structures which are clearly, ears, the nose, the mouth, eyes, etc. However, the shortfalls of the model are also clearly displayed. It can be seen that the model fails to accurately produce finer facial details - such as eyelids, lips, nostrils, and the inner ear structure - with these features instead being noisy and disfigured.

Furthermore, the performance of the model with regard to specific expressions can be seen. There are some expressions which are more accurate to their real world counterpart compared to others. For example, it is clear that the mouth of the *Mouth Open* expression is indeed open and the rest of the face is in correct proportion, while some of the desired proportions of *Cheeks In* such as a clear indent in the cheeks are missing, a difference in quality which corresponds to the metrics of tables 3 and 4. This difference in performance is most likely a result of the model's lack of ability to produce finer details such as the cheek indents of *Cheeks In* which may be more subtle compared to the extreme and broad nature of the features of *Mouth Open*.

Compared to the output of the Simple DC GAN the samples also show a greater tendency to have correct general proportions as there are large and obvious errors such as the collapsed chin in Face 3 of Figure 10.

[8]For a more detailed view, see Appendix A or Video Demo.

It must be noted that there are some errors in these models which are most likely a result of the meshing process such as the hole in the left temple region of the neutral face model which is due to a natural indent in the head which has come too close to the points that form the eye ball, and subsequently the mesh surface has collapsed onto the ball forming the observed hole.

6.3 Comparison against Past Work

As mentioned previously, a number of aspects of the evaluation process were chosen based on similar approaches taken in [4]. This was done with the goal of comparing 3DEC-GAN against the best pre-existing work: MeshGAN and CoMA [20]. As such Table 5 is an excerpt from [4] where Generalisation, Specificity and FID were calculated for both MeshGAN and CoMA.

Table 5: Table from [4] showing calculated metrics as a comparison against CoMA, where the expression control method was used

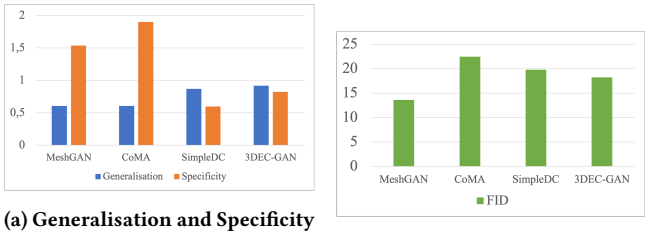
Methods	Generalisation(mm)	Specificity(mm)	FID
CoMA-EXP	0.606±0.203	1.899±0.272	22.43
MeshGAN-EXP	0.605±0.264	1.536±0.153	13.59

When comparing the values in Table 4 against those in Tables 2 and 3 - as seen in Figure 14 - it can be observed that 3DEC-GAN produced lower specificity values but higher generalisation values, compared to MeshGAN. This suggests that 3DEC-GAN produces specific faces that are more accurate considering the training data, but its latent space is more constrained and lacks some ability to produce certain types faces - i.e. it has a weaker understanding of the face in general. This does make sense since MeshGAN learns a 3DMM and therefore functions by manipulating a general 3D model, this allows for finer adjustments which expand the latent space to improve coverage but also leaves it susceptible to unexpected faces which may be some kind of combination, or interpolation, of 2 or more other expressions and/or identities - resulting in the higher specificity result.

Additionally, The FID scores of 3DEC-GAN are similar to that of CoMA and MeshGAN with values that lie between the results for CoMA and MeshGAN. These values suggest that the probability distribution of the output of 3DEC-GANs generator lies closer to that of the training data compared to CoMAs generator but further compared to MeshGANs generator. This indicates a similar ability to produce faces from a visual perspective.

Figure 13: An excerpt from [4] showing faces produced by MeshGAN and CoMA.

However, when comparing the metrics a degree of scepticism must be applied as an inspection of the faces generated by MeshGAN and CoMA, as seen in Figure 13, shows that they are able to produce faces with a much higher degree of accuracy - compared to the output of 3DEC-GAN in Figure 12 - as there is no noise in the faces and all features are in correct proportion. Furthermore, as mentioned in Section 5, the metrics calculations are based on their description in [4] and not taken directly from an open implementation. As such slight differences due to assumptions that needed to be made decrease the accuracy of a direct comparison based solely on metrics.



(a) Generalisation and Specificity results

(b) FID results

Figure 14: Bar graphs showing a summary of the metrics for MeshGAN, CoMA, the Simple DC GAN and 3DEC-GAN

6.4 Discussion

The metrics provided by the evaluation of 3DEC-GAN convey insights into the inner workings of the model. It can be observed through the Generalisation and Specificity values that the model possesses a latent space which, with a moderate degree of success, encodes faces which are expected considering the training data and, importantly, can be sectioned off into regions encoding different expressions. The values for KID and FID further convey the models successes as - based on the images of the generated and testing faces - the inception network was able to determine that the probability distributions of the generator and the real data are relatively close. While these values show that 3DEC-GAN is theoretically promising, the usability and ultimate success of a GAN can only be declared following a visual inspection of the generated samples. It is here that the major failings of 3DEC-GAN can be seen. Figure 12 shows that the faces are too noisy and lack the fine details to truly be a face, despite the fact that it was able to produce general features such as the head itself, ears, the nose, and the mouth to some extent.

Furthermore, this evaluation shows that in comparison to 3DEC-GANs closest counterparts, MeshGAN and CoMA[4, 20], 3DEC-GAN performs similarly well with the regards to quantitative evaluation but again falls short upon visual inspection of the generated samples of all three models. Therefore, while the metrics suggest a comparable performance, the shear lack of usability from 3DEC-GAN disqualifies from being true competitor to this preexisting work.

These findings speak to how well the metrics can be relied upon, as it is clear that while the values produced by a model may appear good, or acceptable, the actual generated samples are not acceptable or not good enough. To this end the metrics must be considered

against a visual inspection of the generated samples and with respect to the subject matter - as the margin of error for a face to be usable and correct is very small.

As such, 3DEC-GAN falls short on its goal of producing usable 3D models, however it does not fail entirely. The generated faces are in proportion with the human head and do display features that are a direct consequence of the desired expression - showing the success of the conditioning step. Should 3DEC-GAN have possessed the ability to generate the missing finer details with less noise, it would be a foremost example of a GAN that works directly on point clouds with no need for highly complex network structures, internal models or mesh operations.

Subsequently, the question remains: what is the reasoning behind these errors in the output, why isn't the model successful? By my understanding there are two reasons for this. Firstly, is the noise itself. Noise is central to GANs; it is the seed from which they produce unique output samples. But noise is also the cause of distortions and artifacts in the output samples[9]. This is why the points themselves do not conform to very strict patterns and why outliers are possible. Secondly, there are errors that are caused by the meshing process and not model itself. Specifically, the methods employed for normal estimation[16] - a process required for the meshing of the face - fail in regions with high point densities, such as the eyes and inner ears. In these regions the normal estimation results in normals pointing in many directions as opposed to uniformly tangent to the face, causing the resulting fuzzy areas as seen in Figure 12. These errors might be corrected by a reduction in the noise in the output samples and/or the inclusion of surface normals in the training data - where the model learns the normals as well as the face, eliminating the need for normal estimation.

3DEC-GAN serves as example of what is possible with these given constraints and the simplicity of the network design - that a deep convolutional GAN can be trained to understand point cloud data and conditioned to allow for control of the generated samples. As such, 3DEC-GAN shows the potential for GANs to generate point clouds in general, not necessarily of faces, a data format that has seen little to no investigation in the domain of Generative Adversarial Networks.

7 CONCLUSIONS

In general, is has been shown that 3DEC-GAN cannot produce adequately usable 3D faces, they simply are too noisy and do not possess certain critical, expression specific, details. To this end the model does not work. However, it is able to produce objects that *resemble* faces and those faces do have features which are a result of the desired expression as per the condition passed into the generator. Furthermore, when submitted to a quantitative analysis, the model performs well, with values comparable to that of the most relevant existing work [4, 20].

While 3DEC-GAN cannot be used in any real world application, it can certainly serve as a base from which to continue work on conditional GANs for 3D face generation and, generally, GANs which produce point clouds of any class of object. It is proof that a GAN is able to understand the overall concept of a point cloud object, with the major shortfalls being found in the finer details. The successes of 3DEC-GAN can be seen in the metrics produced by the model as they show that in a purely mathematical and statistical sense - ignoring the human ability to visually recognise that the output is unacceptable - the model at the very least is succeeding in some right and has the potential to fulfill its full function given additional improvements and refinements.

[9] As seen in [19] where the handwritten digits are "fuzzy" with stray white pixels surrounding the digit structure.

8 FUTURE WORK

Based on these findings it can be seen that 3DEC-GAN best serves as a base from which to conduct further research, or may be used to inform research in similar domains of 3D object generation. To this end there are a number of paths forward:

8.1 Other Objects

The main failure of 3DEC-GAN was in the generation of the finer details which complete a face, but it was able to generate objects which had the correct larger and more general features. Therefore, it stands to reason that a similar GAN may be able to produce a simpler class of objects - where fine details do not have as much of an influence on the success of the model - with a high degree of accuracy.

8.2 More Tuning

While the design and hyperparameters used for the final model were determined to produce the best results through rapid prototyping, out of all variations tested, there are other configurations which were not tested, simply due to time constraints. Therefore further experimentation may yield a better model. Specifically, the areas in which more investigation lies are: the loss function used, the complexity of the networks - such as adding more layers, changing layer size, or adding batch normalisation - and use of a different optimiser.

8.3 Wasserstein Loss and Gradient Penalty

For simplicity sake, the Simple DC GAN and subsequent 3DEC-GAN use a loss function which was taken from the original GAN [15]. However, it is known that if a different loss function based on Wasserstein distance is used, the performance of a GAN can be enhanced, as seen in WGAN [2]. Additionally, a gradient penalty can be included to further improve WGAN, into what is know as WGAN-GP [9]. Both of these improvements were considered but fell victim to time constraints as not only would the new loss function and gradient penalty need to be included, but they would also need to be adjusted to work on our 3D data. These adjustments would require a large amount of further investigation, prototyping and experimentation.

9 ACKNOWLEDGEMENTS

This project involved many shared contributions amongst the research group consisting of Luc van den Handel, Sebastian Oliver and Liam Watson. The project was supervised by Professor Deshen Moodley and is based on his initial project idea of the use of GANs for 3D object generation, with a focus on face generation. Special thanks goes to Professor James Gain, the second reader of this project, who provided key insights on working with 3D objects.

REFERENCES

[1] Hamed Alqahtani, Manolya Kavakli-Thorne, and Gulshan Kumar. 2021. Applications of Generative Adversarial Networks (GANs): An Updated Review. *Archives of Computational Methods in Engineering* 28, 2 (2021), 525–552. https://doi.org/10.1007/s11831-019-09388-y

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.

[3] 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.

[4] Shiyang Cheng, Michael Bronstein, Yuxiang Zhou, Irene Kotsia, Maja Pantic, and Stefanos Zafeiriou. 2019. MeshGAN: Non-linear 3D Morphable Models of Faces. (2019). https://doi.org/10.48550/ARXIV.1903.10384

[5] Li Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142. https://doi.org/10.1109/MSP.2012.2211477

[6] Abdul Fatir. 2018. gan-metrics-pytorch. https://github.com/abdulfatir/gan-metrics-pytorch.

[7] Tom Geller. 2008. Overcoming the uncanny valley. *IEEE computer graphics and applications* 28, 4 (2008), 11–17.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf

[10] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[11] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[12] Nancy Kanwisher and Morris Moscovitch. 2000. The cognitive neuroscience of face processing: An introduction. *Cognitive Neuropsychology* 17, 1-3 (2000), 1–11.

[13] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.

[14] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, Vol. 7.

[15] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. https://doi.org/10.48550/ARXIV.1411.1784

[16] Claudio Mura, Gregory Wyss, and Renato Pajarola. 2018. Robust normal estimation in unstructured 3D point clouds by selective normal space exploration. *The Visual Computer* 34, 6 (2018), 961–971. https://doi.org/10.1007/s00371-018-1542-6

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR* abs/1912.01703 (2019). arXiv:1912.01703 http://arxiv.org/abs/1912.01703

[18] Aladdin Persson. 2020. Pytorch Conditional GAN Tutorial. https://www.youtube.com/watch?v=Hp-jWm2SzR8&ab_channel=AladdinPersson.

[19] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. https://doi.org/10.48550/ARXIV.1511.06434

[20] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. 2018. Generating 3D Faces using Convolutional Mesh Autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[21] Maximilian Seitzer. 2020. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid. Version 0.2.1.

[22] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. https://doi.org/10.48550/ARXIV.1602.07261

[23] Ekin Tiu. 2020. Understanding Latent Space in Machine Learning. https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d. *Towards Data Science* (2 2020).

[24] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[25] Liam Watson. 2022. Controllable 3D Point Cloud Expression GAN.

[26] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. 2017. Towards Large-Pose Face Frontalization in the Wild. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[27] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

A FACES GENERATED BY 3DEC-GAN

For a more detailed view of point clouds and meshes see: Video Demo

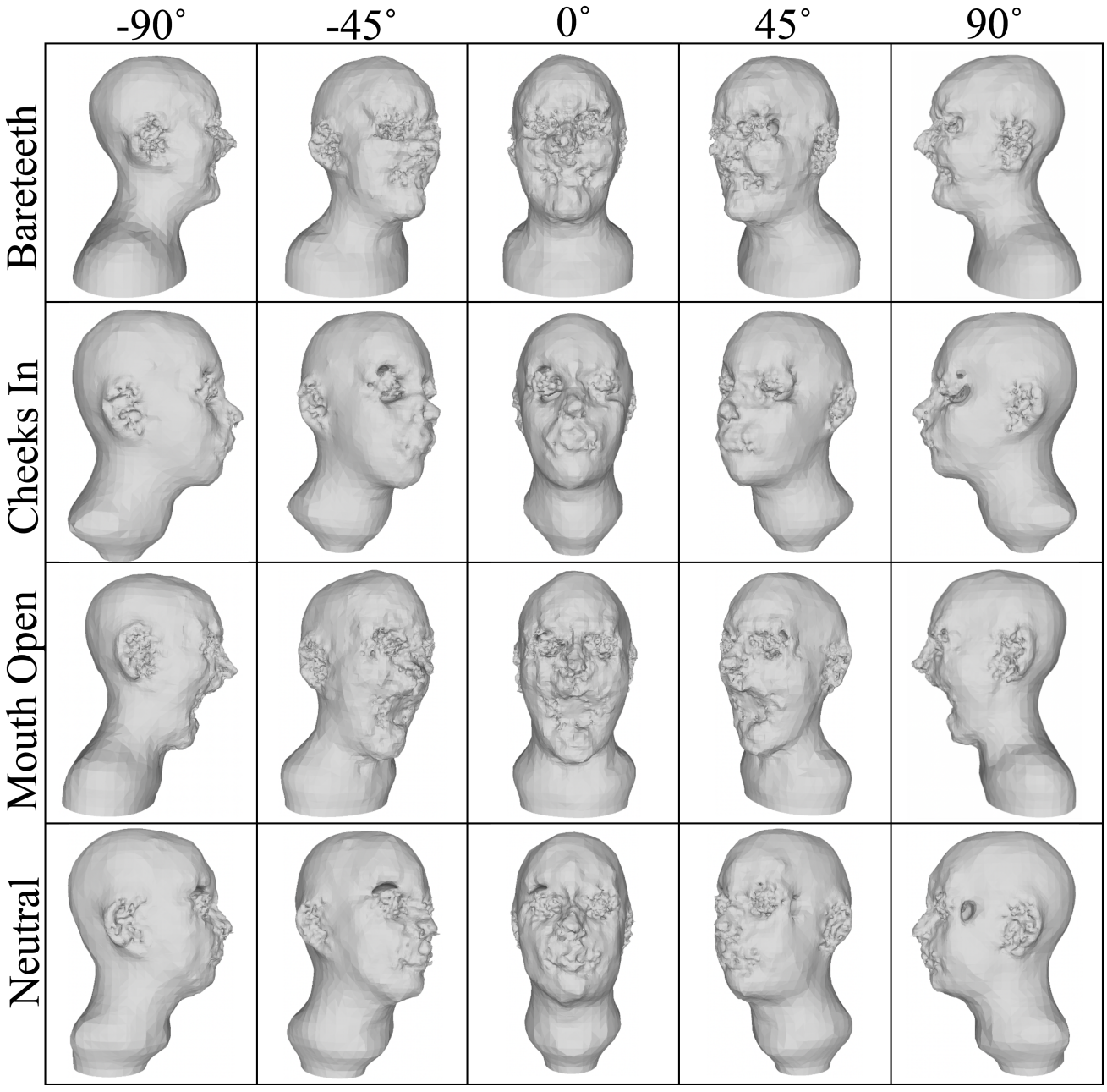


Figure 15: Showing the output from 3DEC-GAN where the noise remained constant while the conditional information vector was varied - i.e. an attempt to generate the same identity in different expressions.

B FACES GENERATED BY SIMPLE DC GAN

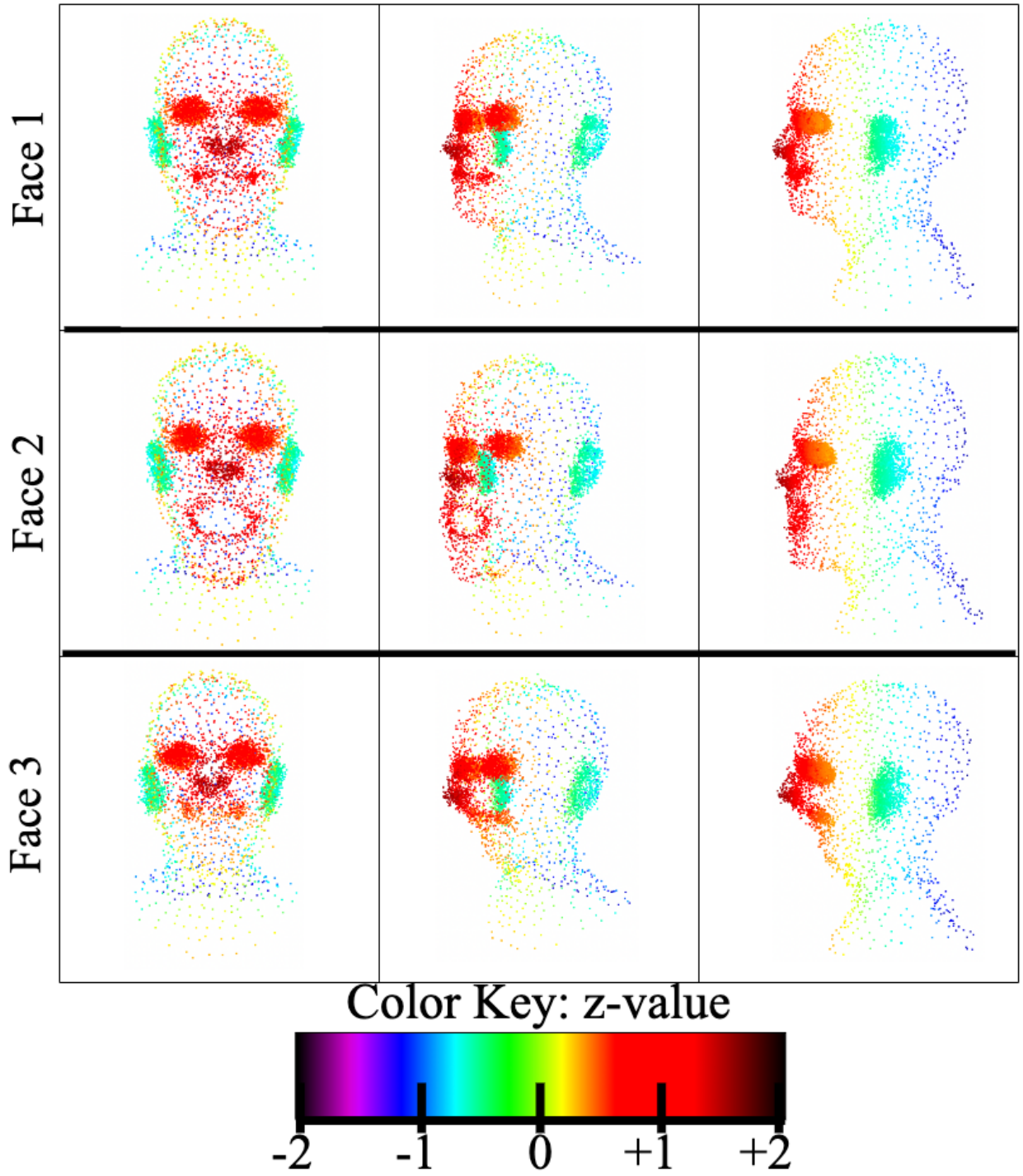


Figure 16: Showing three output samples from the Simple DC GAN, each viewed at the same three angles

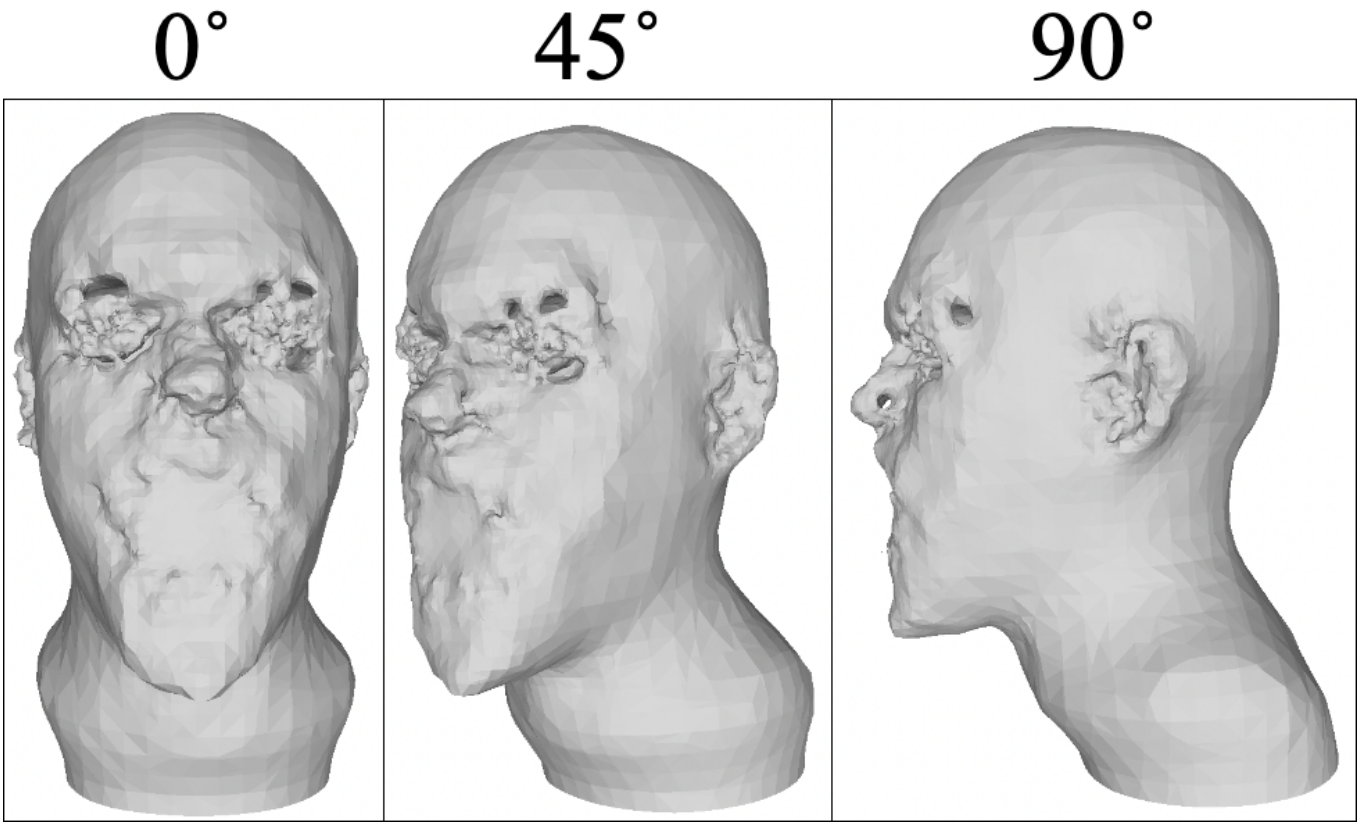


Figure 17: Showing a meshed and rendered sample generated by the Simple DC GAN.

C FACES GENERATED BY MESHGAN AND COMA

Figure 18: An excerpt from [4] showing faces produced by MeshGAN and CoMA.