Creating 3D faces from 2D images using GANs

1st Thomas Reesink
Data Science (DS)
University of Twente
Enschede, Netherlands
thomas@tbjreesink.nl

***Abstract*—This research looks into the viability of generating realistic 3D data based on a single 2D image by means of Generative Adversarial Networks. Many existing methods for generating 3D faces from 2D information require models, making them lose detail as they tend to smooth identifying traits. This research aims to make a system which generates raw 3D data and does not require a predefined model. This is achieved by utilizing Generative Adversarial Networks (GANs) which can generate convincing samples based on a given dataset. By conditioning the GAN it is possible to base the generated 3D data on a given 2D image. In order to objectively measure the quality of 3D data generated by its models, this research trains its models using 3D facial data and uses 3D facial recognition to verify its results. Using 3D facial recognition on the generated samples allows for comparison between methods as well as new insights. By transforming 3D to a 2D matrix it was possible to train a conditional Wasserstein GAN to produce 3D data which could be correctly identified in 63.3% of the cases.**

***Index Terms*—GAN, CGAN, WGAN, 3D, Facial Recognition**

I. Introduction

Computing power has seen improvements in both speed and memory capacity for years. With the improvements in computing power it becomes increasingly simple to consider more variables in problems. Calculations which were very hard or even impossible before are now possible. One variable to consider is 3D, many captures and representations of reality are in 2D, reality however has more dimensions. By adding depth to an image, the image both becomes a more correct representation of reality as well as better understandable for a human.

Adding 3D data has other advantages as well, 3D data is unaffected by orientation or lighting [1]. Making it possible to inspect objects in new ways. However obtaining 3D data is more complex than obtaining 2D data. Most methods for obtaining 3D data are not compatible with simple cameras. Devices which can acquire 3D data and are accurate enough either require the object to remain fixed, have a low resolution or are too expensive.

The goal of this research is to construct a method which generates 3D data based on a 2D image. Generating 3D data from 2D images has been a goal for many researchers for a long time [2]. Many methods were proposed with varying levels of success. However, most of these methods lack a clear and objective performance measurement and rely on visual inspection.

This research focusses on adding 3D data to 2D images of faces. Faces are intrinsically complex and have a lot of detail. This research focusses on the details and identifying traits of the faces. Most methods for generating 3D are used for visualisation or entertainment purposes and do not require a high accuracy, a person inspecting the result would not notice errors as long as the face is consistent. Which is also why the lack of an objective performance measurement is a problem.

To generate the 3D data this research uses a Generative Adversarial Network (GAN) [3]. A GAN is a generative model, generative models aim to generate new and unique samples based on a training dataset. GANs have shown that they are capable to generate very good, almost real samples and is one of the most innovative machine learning structures in recent years [4]. However this research needs the generated samples to be based on a 2D image. To achieve this a condition is added, this structure is called a Conditional GAN (CGAN) [5]. This condition can be a simple class, for example to generate an image of a boat or bridge, but also an entire image.

3D data is often represented as a point cloud. A point cloud is a set of points which have coordinates and possibly other information. Every point corresponds to a 3D measurement and represents an object being present at those coordinates. Point clouds are however not natively compatible with GANs and research is being performed to achieve this [6].

A. Research Questions

To achieve the goal this research will answer the following research questions

- RQ How can we generate 3D faces from 2D images using GANs?
  - RQ1 GANs do not natively support point clouds, how can we make point clouds compatible with GANs?
  - RQ2 A GAN generates new and unique samples, how can we make sure these are based on a given 2D input?
  - RQ3 How can we objectively quantify the quality of generated 3D facial data?

By answering the research questions this research contributed several findings. This research proposes a method to represent 3D point clouds in a machine learning friendly manner which should be compatible with most existing models.

This method preserves most of the 3D data within a predefined range and can be converted from and to a point cloud. Additionally a CGAN based on the Wasserstein GAN [7] called Conditional Wasserstein GAN (CWGAN) was designed which is able to transform data from one domain (2D image) to another (3D data). Within this research the CWGAN was only trained to transform 2D images to the corresponding 3D data, however it could be trained for many other transformations. To objectively measure the performance with a focus on details, 3D facial recognition is used, by comparing the real world performance of the 3D facial recognition with the generated 3D data, the amount of detail retained can be measured.

B. Overview

The paper is structured as such that the following sections first describe the related work is. In Section III the different solutions for the 3D point cloud representation are considered as well as specific considerations when designing a CGAN. The experiments and results section explains the systems and dataset used, after which the 3 experiments and results are described. Finally all conclusions are summarized in Section V.

II. Related Work

In this section, related work is explored. This is done by first researching traditional methods for generating 3D from 2D, these can have important considerations for both handling and generating 3D data. Followed by methods that use deep learning to generate 3D data, GANs are a form of deep learning and thus many findings from deep learning are applicable to GANs as well. After deep learning a general introduction of GANs is given followed by what they can be used for, the problems that GANs suffer from and how these problems can be solved. Finally methods that use GANs in combination with 3D data are explored.

A. Classical 3D from 2D

Though the research to transform 2D into 3D has been going on since the 70' [2] this research will focus on methods specifically made for faces.

Research by D. Jiang et al. [8] showed an efficient way to generate a 3D face from a 2D image. The method they proposed uses a 2D alignment algorithm which was proposed in earlier research by the same group in order to determine key feature points on a 2D image of a face [9]. These feature points are then used to to determine the 3D shape coefficients to create a model. Finally a texture was extracted from the original image. This method however only uses the 3D space to alter the 2D image, the method is able to change pose, change expression and change lighting. The goal of this research is to generate raw 3D data which is usable for methods in the 3D space, the method of D. Jiang et al. only uses the 3D space to generate a new 2D image.

One of the most cited methods is the 3D Morphable Model (3DMM) [10]–[16]. The 3DMM was first introduced in 1999 [17] and later used for facial recognition [18]. A 3DMM is a technique for modelling textured 3D faces. The 3D data of this model consists of an average face with modifier vectors. These modifier vectors map the average face to the actual face based on facial features. Additionally a texture is stored which can be applied to the 3D shape. 3DMM are useful for photo-realistic image manipulations of both orientation and lighting.

In 2003 Romdhani and Vetter constructed a way to generate a 3DMM by means of a single image [10]. They achieve this by using Multi-Feature Fitting (MFF), this fitting combines several features which they define. The features they use are pixel intensity, edge, specular highlight, gaussian prior and texture constraint. For each of these features they defined a cost function. A cost function, or loss function, is a function which scores the output of a method is, the score is then used to optimize parameters. A low loss means the parameters are good. These features are then used to maximise the likelihood of the Multi-Feature Fitting to a face. However research has shown that 3DMM lose a lot of identifying traits and are biased to the average face [19], making 3DMM unsuitable for this research.

A Survey conducted by Levine and Yu in 2009 [20] looked into methods to create 3D facial images from single 2D images. Including the method of D. Jiang et al. [8] and 3DMM itself [10]. They compared four different methods, and found that in order to be efficient D. Jiang et al. [8] sacrifice a significant amount of pixel information, their method uses the data of 87 pixels to estimate the geometry instead of the entire image. However other methods are more dependent on pre-defined models, which as stated have a bias to the average face which makes them unsuitable for this research.

B. Deep 3D from 2D

Since 2011 Deep Learning really took off as graphics processing units (GPUs) were starting to get fast enough to train convolutional neural networks "without" the layer-by-layer pre-training [21]. Because of this, people started to look into new fields where deep learning could be used, generating 3D faces being one of them.

In a research by Tran et al. a method was constructed to generate 3DMM based on a single picture [22]. They noted several interesting things for this research. Known 3D facial datasets are relatively small and therefore not well suited for deep learning. To overcome the lack of training data, Tran et al. used proven methods for generating 3DMM based on multiple images of the same person and assigned the resulting 3DMM to correspond to all used images. By generating new 3D data they were able to artificially increase the training data drastically. However, the accuracy of the new 3D is dependent on the accuracy of the used method.

Tran et al. also noted in the deep learning segment that using a Euclidean loss function resulted in a particular issue. Since 3DMM are based on vectors which belong to a multivariate Gaussian, when using Euclidean loss the resulting generated 3DMM have a bias to the origin, the average face. This happens because those faces on average give a low distance to the desired result and therefore give a low loss. However

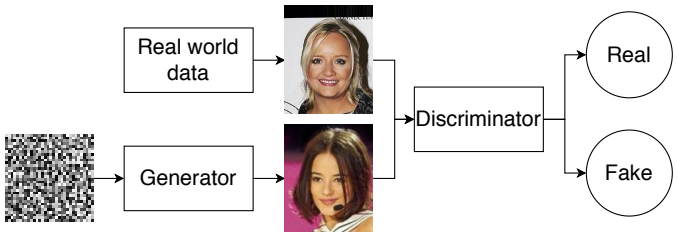

Fig. 1. Generative Adversarial Network

these faces are not unique and not detailed. To overcome this problem they used an asymmetric Euclidean loss function which was designed to encourage the network to favour estimates further away from the origin. They found that the asymmetric Euclidean loss was a great way to prevent the tendency of deep learning methods to produce average results.

Other than Tran et al. many others also tried using Deep Learning to construct 3DMM [23]–[31]. However, as stated before, since 3DMM are based on predefined models and have a bias to the average face, they are not well suited for facial recognition and therefore estimating 3DMM parameters is not the goal of this research.

As far as the author can tell there are no papers describing a method to produce a 3D face from an 2D image through deep learning without the use of a pre-defined model.

C. Generative Adversarial Networks

GANs were first introduced by Goodfellow et al. in 2014 [3]. GANs consist of both a generative model and a discriminative model

Generative models aim to generate the new and unique samples from latent space. Latent space is a hidden space in which observable data can be mapped such that the distance between data is based on similarity. For example in latent space considering shape two chairs are closer to each other than a chair and a table, as the chairs are more similar, however a chair is closer to a table than a bookcase, as they both have four legs and a surface. By generating samples from the latent space between two chairs, a new chair with features from both will be generated. A generative model often takes features from many training samples.

Discriminative models aim to classify their input into given classes. For example in a dataset of animals pictures, a discriminative model can be trained to classify every picture into their species.

Goodfellow et al. noticed the high success of deep discriminative models and the low success of deep generative models. They proposed a way to estimate generative models by simultaneously training two models. Next to training a generative model (generator) they proposed to also train a discriminative model (discriminator) which would decide whether a given input is part of the real dataset or generated by the generator. The generator can then train itself by maximizing the probability of the discriminator making a mistake. The general structure of a GAN can be found in Figure 1.

The generator can be compared to counterfeiters trying to produce fake currency. The discriminator would be the police trying to detect the fake currency. Over time the counterfeiters will get better at faking currency however as better fakes show up, the police will also get better at detecting fakes. In this metaphor the loss of traditional generative models would be comparable to having a perfect police, giving the counterfeiters almost no possibility to improve.

Since their introduction GANs have proven to be valuable to generate realistic samples from a latent space in many different contexts [32]. GANs have been modified to use with images, videos, music, natural languages and medical images. Especially in images, multiple real world applications have been found. Specifically for images, GANs have been used for image transformation, super-resolution, object detection, object transfiguration, face transformation, image generation, image restoration and brain activity visualisation [4], [32], [33]. Lastly GANs have empirically shown to produce more realistic results than other generative models [32].

D. Problems with GANs

Even though GANs have many advantages and possibilities, they suffer from instability. This is among other reasons because the generator and discriminator do not know when they have reached the ultimate strategy. When the generator is making "perfect" fakes, very close the actual dataset, the discriminator will try to root them out, to reduce False Positives, and by doing so it classifies more real samples as False Negative. Therefore the generated results of a GAN can get worse over time, making it prone to instability [32].

Another problem with GANs is Mode collapse. Mode collapse is the phenomenon when a generator does not generate all different kinds of samples present in the original dataset. A mode, or kind of sample, is a subsection of the training dataset. For example, in the MNIST dataset [34] which contains handwritten numbers, see Figure 2, all numbers fall into the training dataset and the subset of all zeros can be considered a mode. Since the only goal of the generator is to fool the discriminator, if it trains itself to only produce one mode which is good enough, it achieves its goal but not the real goal. In the example mode collapse could occur where only the number four would be generated, or when all numbers except the number nine are occurring.

E. Wasserstein GAN

One way to overcome mode collapse was proposed by Arjovsky et al. [7]. They called it the Wasserstein GAN (WGAN). In a WGAN instead of having a discriminator decide whether a given input sample is real or fake it only has a continuous value output. This allows for a more gradual approach to updating weights. In the paper they compare this discriminator to an art critic, it won't decide whether it is real art, it will decide whether it is good art by giving it a grade. Arjovsky et al. state that, mode collapse comes from the fact that the optimal generator for a fixed discriminator is a sum of deltas on the points which the discriminator assigns

Fig. 2. Samples from the MNIST dataset [34]

the highest values. For example, in an MNIST experiment the discriminator gives a specific shape of the number nine a high probability to be real. The discriminator updating is paused and the generator then gets trained till it has a very low loss. Most generated samples will look exactly like the shape of the number nine with the high probability and the other modes will not show up as this is what the loss of the generator seeks to achieve. Because the loss of a WGAN is based on a grade instead of a choice, it can lower the grade of often generated modes without being wrong, which would increase its loss. Additionally the modes that are seldomly generated, gradually get a higher grade. This way mode collapse will not occur according to Arjovsky et al..

WGAN uses Earth-Mover (EM) distance, aka Wasserstein-1, to determine its loss. EM distance is a representation of the distance between two probability distributions within a specific region. Thus instead of looking at the distance of a real sample and a generated sample, EM considers the difference between the distribution of real samples and the distribution of generated samples. EM distance has a Lipschitz constraint which needs to be guaranteed. A Lipschitz constraint limits the slope of a continuous function. Because of this the update of the weights has to be within a specific range. In the WGAN paper they achieve this by means of weight clipping, however they state that this is "a clearly terrible way to enforce a Lipschitz constraint". Gulrajani et al. improved the WGAN by replacing the weight clipping with a gradient penalty [35]. The new method has a more complex loss function. This results in a longer training time per batch. However they also show that the results after the same real world time are better than those when using weight clipping.

F. Conditional GANs

Conditional GANs (CGANs) were already proposed by Goodfellow et al. in the original GAN paper [3]. In a research by Mirza and Osindero [5], they explain how to implement a condition as well as show the potential. A conditional GAN takes the completely random generating GAN and conditions

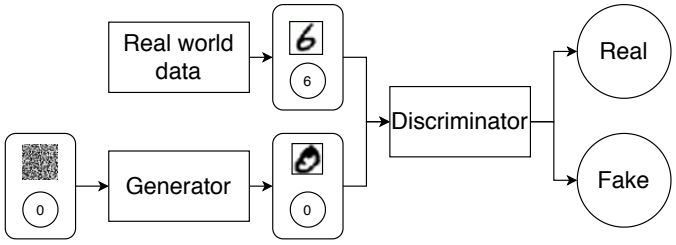

Fig. 3. Conditional Generative Adversarial Network

both the discriminator and generator on some kind of conditional information. In the paper they used the MNIST dataset (Figure 2) along with the classes (0-9) which are also in the dataset as a condition. This allowed them to force the generator to generate a specific number instead of a random number. The structure they used can be found in Figure 3. Though this is a simple example, they state that the conditional information can be any kind of auxiliary information. The conditioning is performed by adding the conditional information by feeding it into both the discriminator and generator as additional input layer.

G. GAN 3D

Due to the potential GANs have shown in various fields, researchers have also sought to implement the method for experiments with 3D data.

Following the research of Tran et al. [22] Galteri et al. saw an opportunity to improve the results by using GANs [36]. They state that generating 3D where no a priori knowledge is available about the image is challenging. Therefore they decided to first generate a 3DMM and use a Conditional GAN (CGAN) to improve it. To make the 3DMM compatible with a CGAN they convert it to three separate 2D matrices, based on the research by Gilani et al. [37]. This method loses no information compared to the 3DMM. These matrices are then used as inputs for teaching the CGAN.

They compared several different models with different inputs for the Generator, Discriminator and Loss function. One of the matrices was based on depth. They found that reducing the amount of information for the discriminator and loss function to only the depth matrix actually lowered the mean average error between the ground truth and the estimate. They explain that this is because when the depth of a face is coherent the other data is inherently also coherent.

Because most methods for 3D representations based on 2D images are based on models (see section II-B) they are not suitable for this research, as the goal is to be able to use the generated data within 3D space. S. Moschoglou et al. proposed a method to generate 3D faces as a 3D mesh [38]. They also state no GAN-based method has been proposed in the literature that can successfully represent, generate or translate 3D facial shapes(meshes).

The method they proposed converts the 3D face meshes to UV spatial maps [39], this way the face data is represented in 2D space and easier to put into a network. An UV spatial

map is a 2D image recording the 3D coordinates of a complete facial point cloud. It is a depth image with semantic meaning at each pixel. The method uses an encoder-decoder structure for both the Generator and Discriminator. An encoder-decoder structure first lowers the amount of channels (encoding) and then increases the amount of channels (decoding) to the original amount. This forces the model to focus on rough features over details. They claim means that the encoder-decoder models can be pre-trained before the adversarial training.

Though the research of Galteri et al. [36] and S. Moschoglou et al. [38] are cutting edge, they do not look into a way to directly base the 3D face on a 2D image. As far as I can tell, no-one has attempted to construct a 3D face based on a single image using GANs.

One of the biggest problems, preventing using 3D data in combination with deep learning, is the size of the datasets. Tran et al. have stated that the facial datasets are relatively small for deep learning [22]. As GAN is based on two Networks that use deep learning, it suffers from the same problems. In a dataset reduction research by Fajar Ulin Nuha et al. [40] they also confirm that datasets of around 2000 samples give poor results, they recommend around 50,000 samples. Other research specifically focussed on faces has shown good results with around 14,000 samples [41]. The most popular datasets like FRGC v2.0 [42] (4,007 3D images), Bosphorus [43] (4,666 3D images) and Casia 3D[1] (4624 3D images) are all too small. As such it is needed to combine multiple datasets or augment the dataset. Augmentation can be done for example by rotating faces [41], mirroring or adding noise.

III. Methods

This research aims to generate 3D facial data from 2D facial data using GANs. In order to achieve this, a 3D presentation compatible with most GANs as well as a state of the art Conditional GAN were designed.

A. 3D Representation

The 3D datasets stated in Section II-G consist of point clouds, a list of x, y and z coordinates. GANs have a fixed number of inputs and outputs. Thus it is not possible to directly output a non fixed amount of points. With 3DFaceGAN S. Moschoglou et al. [38] already showed that it is possible to generate 3D data using GANs. However there are other possible methods.

The biggest challenge to overcome is the multidimensional aspect. Possible methods to process this data multidimensional data in a GAN are:

- x, y and z coordinates as nodes for every pixel, as shown in Figure 4
- a separate generator for every dimension with one discriminator, as shown in Figure 5
- Construct a single layer representation, focussed on depth, as shown in Figure 6

[1]CASIA-FaceV5, http://biometrics.idealtest.org/

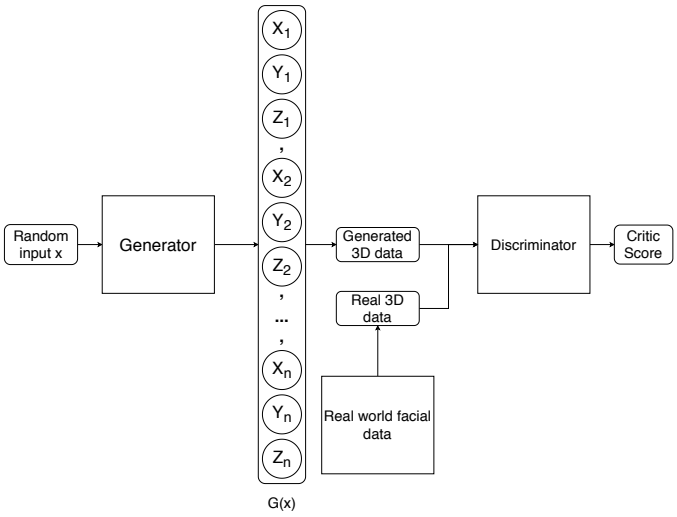

Fig. 4. Simple WGAN

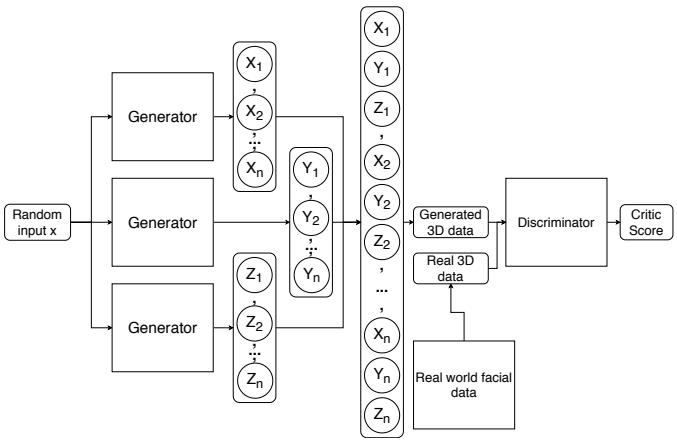

Fig. 5. Split generator WGAN

This research uses the third option, a depth image (Figure 6). The x and y coordinates are encoded as part of the coordinate within the depth image. This way the resulting images can be easily transformed to real world data. By representing the 3D data in a 2D matrix the data becomes compatible with most GANs. A plane is selected on which the 3D is placed, from a frontal view the depth compared to this plane is recorded in the 2D matrix. The methods from Figures 4 and 5 can be considered for future work.

In Figure 7 a sample registration can be found. The registration is performed using an frontal projection to a 2D plane. In this plane the depth is stored in mm in a grid of 165x195. The real world distance between the pixels is fixed, 0.67 mm, this ensures that 3D points can be calculated. The position of the face on the plane is determined based on the nose, this ensures that the facial features are in roughly the same spot for all samples. Additionally the 2D images are registered to have a grayscale value corresponding to the 3D measurement.

In order to lower the training duration as well as increase the chance that the GAN successfully generates faces the methods

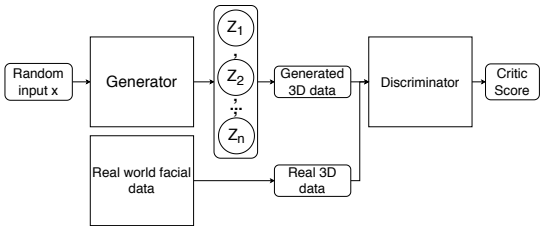

Fig. 6. Only depth WGAN

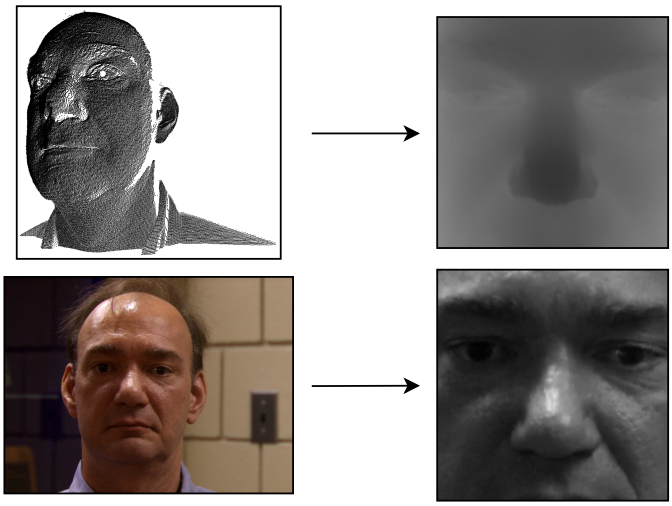

Fig. 7. Registering of 3D and 2D data

are first confirmed at lower resolutions. To accomplish this the above data has been rescaled and cropped to 32x32, 64x64 and 128x128 resolutions. To construct these resolutions bilinear interpolation is used, since it can be assumed that the skin is continuous and roughly linear between known datapoints, this should give a decent interpolation. In order to improve the compatibility with known models the depth data was transformed to be in the [-1, 1] range.

B. Conditional GAN

A regular GAN is purely trained to generate new samples, the goal of this research however is not to generate new 3D faces but to generate 3D facial data which corresponds to a 2D image. This is achieved by adding a condition to the training of the GAN. The condition in this case being a 2D image. This is done by using the conditional GAN (CGAN) method as first described by Mirza et al. [5], a schematic can be found in Figure 8.

The CGAN has a generator which has an encoder-decoder structure in order to transform from an 2D input to corresponding 3D data. This way the model is forced to focus more on how pixels are related rather than every pixel individually.

The generator of a regular GAN uses a random vector (aka noise), which it maps back to the image space. The output is then presented to the discriminator. By using this noise in combination with the trained weights a GAN is able to generate the samples. Because there is an actual input for

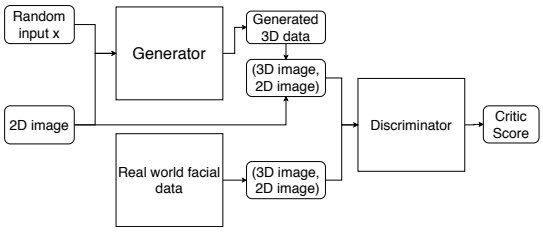

Fig. 8. Conditional WGAN

the generator of a CGAN (the 2D image), the random vector becomes optional. Various research have removed the random vector input of CGAN [44]–[46], in this research both were implemented and compared. The random noise is added as an additional input layer of the same size as the original image.

The possible models are:

1) Generator with only 2D input
2) Generator with both 2D and noise input

IV. Experiments and Results

This research answers its research questions by performing three experiments.

Experiment 1 focusses on the question posed in RQ1, how 3D data can be adapted to be used in GANs, the product is a GAN that can create 3D facial data from latent space. Thus experiment 1 is to select and train GANs with the prepared data.

Experiment 2 will add the conditional information (2D image) as an input for the GAN in order to answer RQ2. The product is a Conditional GAN that can create 3D data corresponding to an 2D image.

Experiment 3 is a verification experiment to answer RQ3, it uses known 3D facial recognition on generated 3D samples to compare them to their ground truth.

A. Systems

All machine learning is done in Python 3.7, this choice was made due to the experience of the author and the available libraries that support machine learning in Python. Version 3.7 was chosen as it is the most recent stable release at the start of the research. All machine learning models were defined in Pytorch 1.3.1. Pytorch was recommended and version 1.3.1 was the most recent version at the start of this research.

The known 3D facial recognition for Experiment 3 was produced by Spreeuwers in 2011 and is called FaceUT3D [47], [48]. This method was chosen as it has a high identification accuracy and collaboration with the creator is possible.

B. Dataset

The primary dataset used in this research is the FRGC v2.0 dataset. FRGC is a dataset of 3D facial data with corresponding 2D information that was released to promote and advance face recognition technology [42]. It contains 4,007 3D images of 466 persons. The 3D data is represented as a grid of 480x640 3D points. The corresponding 2D data is a pixelmap of 480x640 24 bit color pixels.

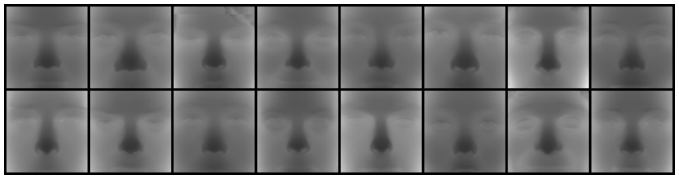

Fig. 9. Sample batch from the 3D data of FRGC v2.0 after preparation

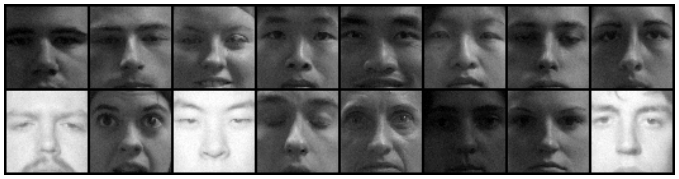

Fig. 10. Sample batch from the 2D data of FRGC v2.0 after preparation

This dataset has been converted to registered depth images described in section III-A. A sample batch of the prepared images can be found in Figure 9 and Figure 10. In order to separate the dataset in a train and test set, 10% of the persons are designated as testpersons, the remaining persons are the training set.

This dataset is too small as discussed in Section II-G. It was not possible to combine the datasets in time for this research as the different datasets have different structures which are not completely compatible. Training with a combined dataset should be considered future work. In order to augment the data, Gaussian noise was added. This research compares various intensities of Gaussian noise on the 2D images in order to augment the dataset.

There were 4 different noise levels to find which amount of Gaussian Noise on the 2D input gave the best results. The datasets used are:

1) No noise
2) Gaussian Noise, mean=0, var=0.001
3) Gaussian Noise, mean=0, var=0.002
4) Gaussian Noise, mean=0, var=0.005

A sample of each of these noise levels on different image sizes can be found in Figure 11.

Additionally other methods for augmenting data were considered. Flipping the image and 3D data in the y direction should provide valid faces and double the dataset. Selecting multiple crops of the face was also considered, however this method could also have negative effects by moving feature locations within the image could make it harder to train. Both were considered but not used due to time constraints, there was not enough time to thoroughly compare the different results.

C. Exp 1: Wasserstein GAN with Gradient Penalty

In order to test whether the 3D data representation described in Section III-A works a GAN was trained.

The GAN used in this research was selected by a preliminary experiment on the CelebA dataset [49]. CelebA contains images of 2D faces, making it relatively close to the actual data in this research. The considered GANs were DCGAN [50],

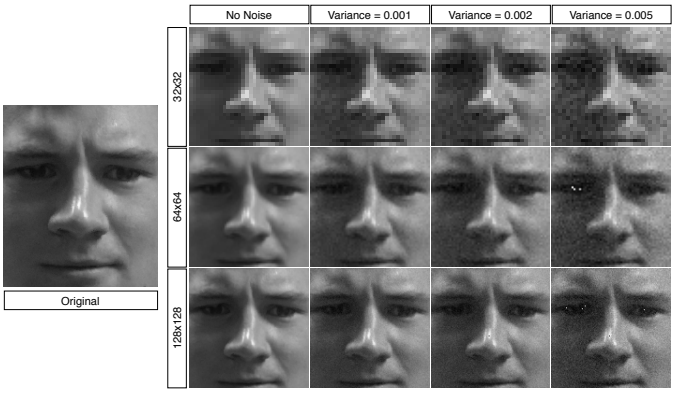

Fig. 11. Comparison of how different noise levels affect the used image sizes

TABLE I
WGAN TESTBATCHES

Testbatch	Model	Dataset	Resolution	Iterations
t1	WGAN	Registered FRGC	32x32	35,000
t2	WGAN	Registered FRGC	64x64	70,000
t3	WGAN	Registered FRGC	64x64	70,000

WGAN [7] and WGAN with Gradient Penalty (WGANGP) [35]. Wasserstein GAN with gradient penalty was chosen as it produced the most unique results without defects, is state of the art and claims to have solved mode collapse.

The specific layers that were used for this experiment can be found in Appendix A. The WGAN was trained with the images of the 32x32 dataset as well as the 64x64 dataset. The entire dataset was used as the focus was to create new and unique samples there was no need for a comparison between seen and unseen data. All training was done with a standard batch size of 64 as given by Arjovsky et al.

The goal of a generative model is best described as to generate samples with variation which have not been seen before. Since samples which exist in the sample database do not need to be generated and a model that generates roughly the same sample every time is not very useful. To analyse these two factors three generated sample batches are taken, one from the 32x32 trained model (t1) and two from the 64x64 trained model (t2 and t3), an overview of these batches can be found in Table I. The t1 and t2 are visually inspected whether there is variation in the results. In order to check confirm whether the generated samples are new and not part of the original dataset, t3 has the closest match within the original dataset looked up and the absolute difference displayed. If all comparisons show differences then the experiment can be considered a success.

A sample batch generated by the WGANGP trained on 32x32 data, t1, can be found in Figure 12, these samples look promising and show variation. The variation is most obvious around the nose, there is also variation in the eyes and some samples even have hair.

In Figure 13 a sample batch generated by the WGANGP trained on 64x64 depth images can be found, t2. In these samples the difference is also most noticeable around the nose. The higher resolution makes the width of the nose more

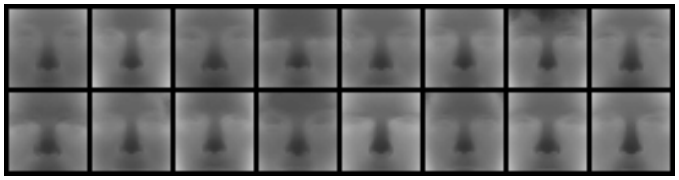

Fig. 12. Testbatch t1: Generated samples by WGANGP after 35,000 training batches of the prepared 32x32 dataset

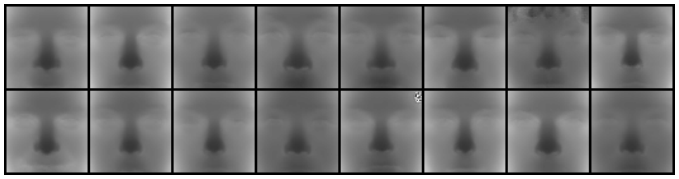

Fig. 13. Testbatch t2: Generated samples by WGANGP after 70,000 training batches of the prepared 64x64 dataset

visible. The variation around the eyes seems lower however is still there. The variation in hairstyle is also still present. Additionally there is now a sample with a visible mouth (second row first sample). Unfortunately one sample has an artefact(second row fifth sample), artefacts should not present. Upon inspection of 640 samples(10 batches) 11 samples had artefacts, 1.7%.

In Figure 14, t3 is displayed along with the closest matches. By portraying the closest match alongside a sample it is possible to determine whether the sample is actually new or that the WGANGP is generating existing faces. In the first sample(first row first sample) the main difference is in the eyebrows as the match has them higher, in the second the nose of the real sample is more upright and in the third sample the nose in the real sample is less pointy. In the difference row it is clear that every match has some difference.

As there are differences between all images, some clearer than others, it can be concluded that there are new and unique samples generated.

As the WGANGP was able to generate new and unique faces with variation, the 3D representation proposed in Section III-A is a successful method to represent 3D data for a GAN. Thus the proposed method is a valid solution for RQ1. The artefacts should not have been present but do not invalidate the above conclusion, they are probably a result of the small dataset size.

D. Exp 2: Conditional WGAN

The core of this research is to generate 3D faces from 2D images. The previous section showed that a WGANGP using the 3D representation is able to generate 3D data. In the second experiment the WGANGP was altered to accept an image as described in Section III-B to create the Conditional WGAN (CWGAN). The full structure of the used CWGAN can be found in Appendix B.

As described in Section B there are two models, one with and one without noise as input. Additionally to augment the

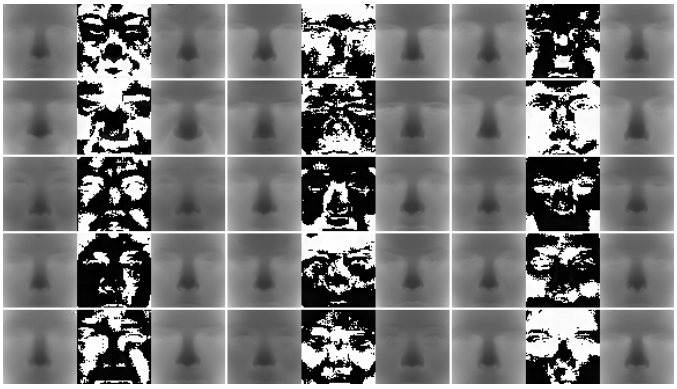

Fig. 14. Testbatch t3: Generated samples (left) by WGANGP after 70,000 training batches of the prepared 64x64 dataset with the closest match (right) in the original dataset and the difference (middle). The difference is from low(black) to high(white).

TABLE II
SCENARIOS AND THEIR PARAMETERS

Scenario	Model	Dataset Augmented	Variance
CWGAN-NoNoise	CWGAN	No	-
CWGAN-0.001Var	CWGAN	Yes	0.001
CWGAN-0.002Var	CWGAN	Yes	0.002
CWGAN-0.005Var	CWGAN	Yes	0.005
CWGAN_NOISY-NoNoise	CWGAN_NOISY	No	-
CWGAN_NOISY-0.001Var	CWGAN_NOISY	Yes	0.001
CWGAN_NOISY-0.002Var	CWGAN_NOISY	Yes	0.002
CWGAN_NOISY-0.005Var	CWGAN_NOISY	Yes	0.005

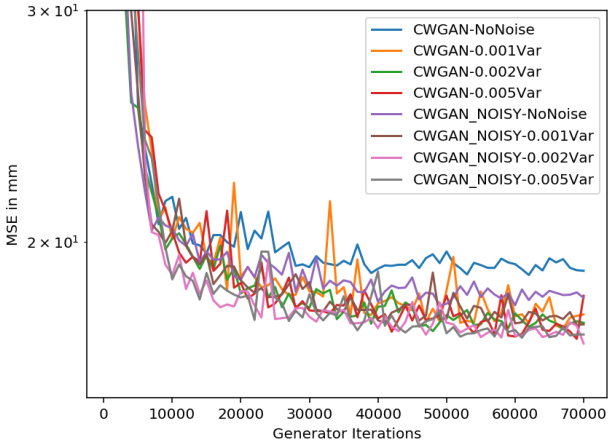

Fig. 15. MSE error between generated samples by several CWGAN and ground truths from the unregistered 2D 64x64 dataset

dataset, noise is added to the 2D data as described in Section IV-B. This noise is randomly assigned every time a sample is used, thus independent of image size and amount of samples in the dataset. This results in 8 scenarios shown in Table II.

When comparing the performance of a conditional GAN there are additional objective performance measurements. These are possible as the ground truth is known in this case. As such the CWGAN is only trained on the trainpersons and

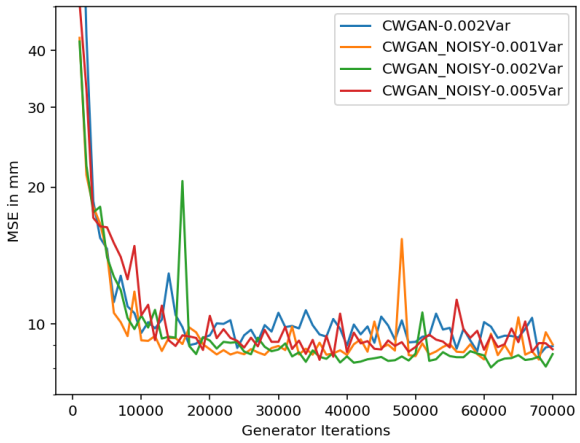

Fig. 16. MSE error between generated samples by several CWGAN and ground truths from the registered 2D 64x64 dataset

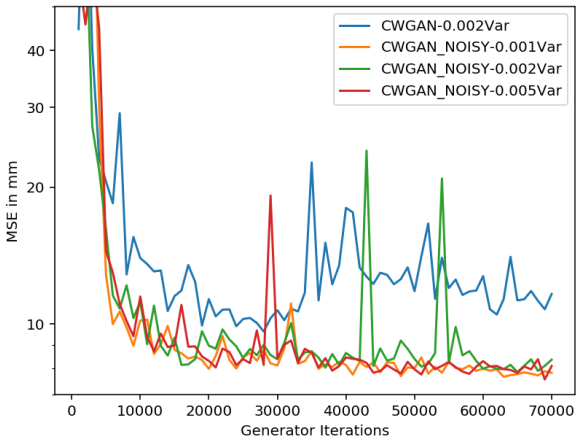

Fig. 17. MSE error between generated samples by several CWGAN and ground truths from the registered 2D 128x128 dataset

the objective measurements are based on the testpersons.

To estimate the best model and noise variance, the mean squared error (MSE) between the generated samples of a scenario and the corresponding ground truths is calculated for all test persons. The reasoning is that a lower MSE should indicate that the the generated sample should be closer to the wished result. The trained models for this estimation were based on unregistered 2D data with registered 3D data both with a resolution of 64x64.

In Figure 15 the progression of the MSE between generated samples and the ground truths is displayed. It is clear that the MSE when not augmenting the data (no noise) is the highest. It also seems that the CWGAN_NOISY model is more stable, the CWGAN model has more and higher spikes. In the end however most seem perform equally well.

The four scenarios with the lowest MSE after 70,000 iterations were selected for additional dataset training. These are, in order from lowest to highest MSE:

1) CWGAN_NOISY-0.002Var
2) CWGAN_NOISY-0.005Var
3) CWGAN_NOISY-0.001Var
4) CWGAN-0.002Var

The additional datasets used for training are based on registered data for both the 2D and 3D data, as shown in Figures 9 and 10. By reducing the amount of transformation that the generator needs to learn, by fixing the 2D datapoints to the corresponding 3D datapoints, the resulting model should be able to give better results. By increasing the resolution of the image the error may increase, intuitively as the generative model needs to generate more data training is difficult.

By registering the the 2D information the MSE in the result was lowered by around 8 mm, this is a lot as it nearly halves the MSE. Increasing the resolution had the expected result on CWGAN_0.002Var however on the CWGAN_NOISY the resulting error did not increase. Though all CWGAN_NOISY models ended clearly below CWGAN_0.002Var however both CWGAN_NOISY_0.002Var and CWGAN_NOISY_0.005Var had more extreme spikes The MSE of CWGAN_NOISY_0.001Var was the most stable of all models. It is clear that the model with noise performs better when training at higher resolutions than the model without noise.

Based on these results it can be concluded that a GAN is able to generate 3D data based on 2D images, however the fact that the generator is able to transform data from one domain to another does not mean that the generated faces actually matches with the person in the 2D data, just that a set of 3D facial data is made with 2D images as input. It can also be concluded that the augmentation of the dataset helps to lower the distance between generated samples and the ground truth. What cannot be concluded is that augmentation will always help, only that it helps here. It is possible that the positive effect of data augmentation is because the dataset has a small size. Lastly it can be concluded that removing the noise from the generator when using an image as generator input is not beneficial for the results.

E. Exp 3: 3D facial recognition

As stated in the previous section, MSE only checks the similarity between two images on a surface level. MSE focusses on a overall depth distance instead of identifying treats and thus it could not be concluded that the generated facial data matches the person it is based on. To measure this performance a third experiment was performed. In the third experiment the resulting CWGANs from experiment 2 is verified by means of 3D facial recognition.

The 3D facial recognition method used was designed by Spreeuwers [47], [48]. This method produces a similarity score between 0 and 60 on the test and reference data. These 60 points are based on 60 “judges” which score either 0 or 1 on different regions. These judges are set to have a threshold

which corresponds with a maximum False Acceptance Rate of 10%. Resulting in an identification rate of 99.3% and verification rate of 99.4% at FAR=0.1% on real data.

For the experiment several datasets were generated using the models trained in the previous experiment. CWGAN_NOISY_0.001Var and CWGAN_0.002Var are examined in depth as these performed the best for CWGAN_NOISY and CWGAN respectively. For every sample in the original dataset for every scenario 3D data was generated. So for every sample the scenario datasets contain:

1) 2D image, real2D
2) 3D data, real3D
3) generated 3D data, fake3D

The fake3D samples can be considered as a new 3D measurement and are used as probes, the real3D samples are used as the gallery. This results in a score between 0 and 60 for every fake3d, real3d combination. There are two types of tests present in this scoring. Genuine test, when the fake3D data is compared to the real3D data of the same person. Imposter test, when the fake3D data is not from the same person as the real3D data it is compared to. The CWGAN is performing properly if the genuine tests score observably higher than the imposter tests.

There are 2 considerations, a genuine test should score as high as possible and a imposter test should score as low as possible, though similar these do not consider the same thing. The imposter score can be correlated to the similarity of faces, thus if the model generates an average face the imposters score higher. The genuine score can be correlated to the uniqueness of faces, thus how well the model is able to transform identifying features. The less the genuines and imposters overlap in score the better our generated faces are.

To visualize the above considerations the scores are displayed in a normalized bar graph on a logarithmic scale. Additionally the identification and verification rates are calculated. The identification rate is the percentage of cases where the highest FaceUT3D match score belonged to the correct person. The verification rate is the percentage of genuine persons which pass whilst only allowing 0.1% of the imposters to pass the check.

From the MSE results in Figures16 and 17 it is expected that CWGAN_002VAR and CWGAN_NOISY_001VAR score roughly the same on 64x64 images, CWGAN_NOISY_001VAR performs roughly the same on 64x64 and 128x128 images and that CWGAN_002VAR scores worse than CWGAN_NOISY_001VAR on 128x128.

In Figures 18 and 19 a normalized histogram of the scores by genuine and imposter tests. Figure 18 and Figure 19 look mostly identical, however CWGAN_NOISY_001VAR seems to have higher scores in the 50-60 range.

Extending the same comparison to 128x128 images, an interesting phenomena occurs, Figures 20 and 21. More than half of the imposters score zero. This can probably be accounted to the FaceUT3D method working better with more data points.

Figure 21 shows the same as Figure 19, namely that CWGAN_NOISY_001VAR has more genuines with higher

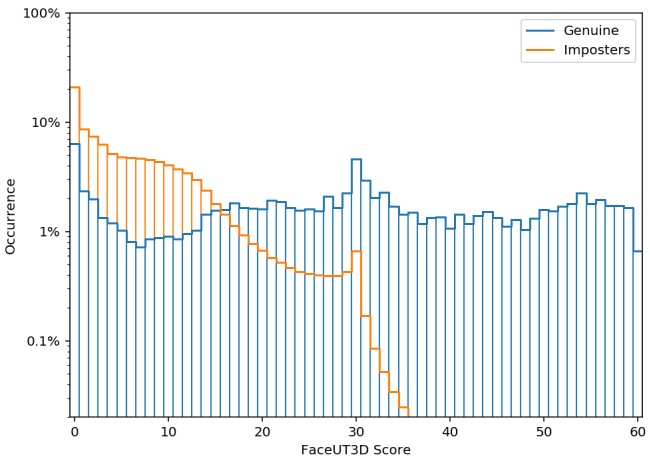

Fig. 18. Comparison of scores by genuine and imposters normalized for the amount of cases for CWGAN_002VAR with 64x64 images

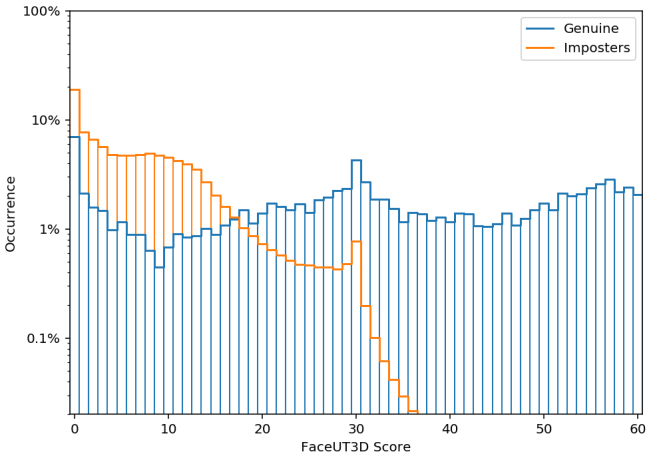

Fig. 19. Comparison of scores by genuine and imposters normalized for the amount of cases for CWGAN_NOISY_001VAR with 64x64 images

scores and less genuines with zero score compared to CWGAN_002VAR. However imposters also score higher in CWGAN_NOISY_001VAR, in both models most imposters score below 31. The higher score of CWGAN_NOISY_001VAR is in line with the expected result.

The values in Table III give an overview of all the model results. CWGAN_NOISY_001VAR scored the highest for both 64x64 and 128x128 images, and CWGAN_002VAR scored the lowest for both 64x64 and 128x128 images. Whilst the identification rate of CWGAN_002VAR on 64x64 is roughly equal the verification rate is about 5.0% lower than the others. This means that even though the samples are roughly as good, they are more similar. The 128x128 results show that even though the MSE loss of CWGAN_NOISY_001VAR on 64x64 was lower than on 128x128, the verification rate is lower. As stated before, a lower verification rate is correlated to more similar faces, which is unwanted. The highest identification score of CWGAN_NOISY_001VAR is in line with the expected result.

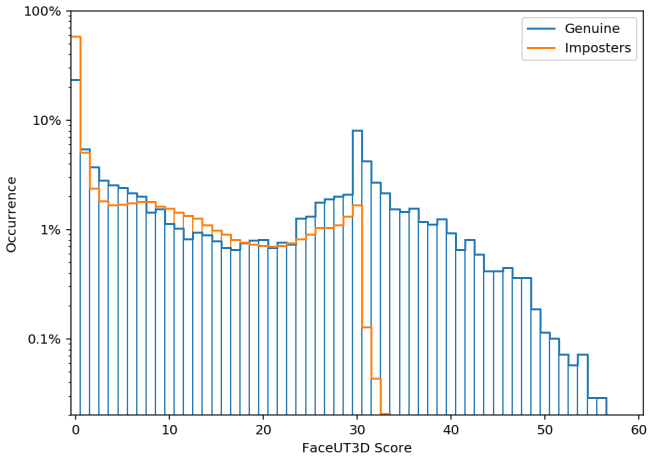

Fig. 20. Comparison of scores by genuine and imposters normalized for the amount of cases for CWGAN_002VAR with 128x128 images

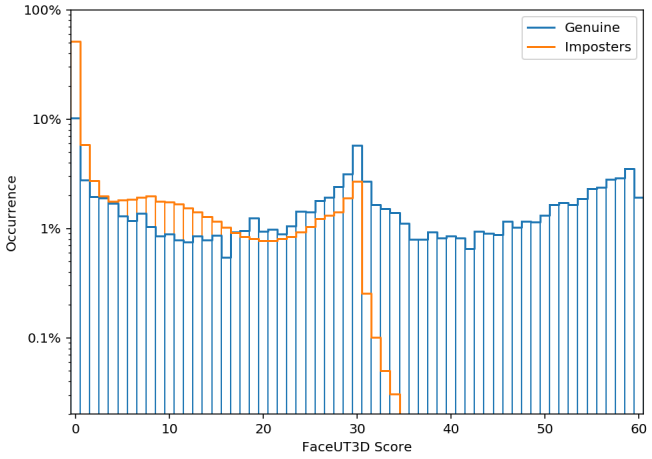

Fig. 21. Comparison of scores by genuine and imposters normalized for the amount of cases for CWGAN_NOISY_001VAR with 128x128 images

Based on that imposters score lower than genuines and a identification rate of 63.3%, it can be concluded that the CWGAN is able to determine identifying treats of faces and transform them to actual 3D data. However due to the high overlap in the scores up to 30 the method is not directly viable, as seen in the verification rates. These results could probably be increased by collecting a larger dataset.

Using 3D facial recognition to verify generated 3D faces has given good and objectively comparable results. It can be concluded that 3D facial recognition is a good way to verify and measure the performance of a 2D image to 3D face method. This method of comparing 3D faces should be considered for research which generate 3D facial data.

Additionally it can be concluded that removing the noise from the CWGAN generator gives measurable worse results, especially on higher resolutions.

TABLE III
IDENTIFICATION AND VERIFICATION RATES OF THE DIFFERENT MODELS

Model	Data	Identification rate	Verification rate at FAR=0.1%
CWGAN_002VAR	64x64	57.6%	36.5%
CWGAN_NOISY_001VAR	64x64	58.6%	41.6%
CWGAN_NOISY_002VAR	64x64	56.3%	41.6%
CWGAN_NOISY_005VAR	64x64	57.3%	40.2%
CWGAN_002VAR	128x128	49.6%	18.6%
CWGAN_NOISY_001VAR	128x128	63.3%	39.4%
CWGAN_NOISY_002VAR	128x128	59.6%	38.9%
CWGAN_NOISY_005VAR	128x128	59.6%	37.2%

V. CONCLUSION

This research aimed to improve facial recognition. Most facial recognition only consider 2D images, mostly because this is easier to acquire than 3D data. Instead of using measurement devices this research looked into a method to generate the 3D data from 2D using machine learning. With an additional focus on usability in 3D facial recognition.

This research answers the question, “how can we generate 3D faces from 2D faces by using GANs?”, by answering three sub questions.

A method to represent 3D data which is compatible with most GAN and other generative models was proposed which is able to preserve most of the 3D information. This method was verified by successfully training a WGAN to generate unique and varying 3D faces using this method.

To base the generated 3D faces on 2D images, a conditional WGAN was designed to accept both a 2D and random input. This research has shown that removing the random input of the WGAN produces measurably worse results, especially on higher resolutions. Based on a mean squared error of less than one mm, it was concluded that this CWGAN transforms 2D images to 3D data. Thus that adding a condition as first designed by Mirza et al. [5] a 3D face can be based on a 2D image.

In order to verify that the generated faces objectively belong to the person they are based on, this research proposed to use 3D facial recognition. By using 3D facial recognition it was possible to see that the CWGAN preserves at least some of the identifying features of the faces. By comparing the identification rate, a method can be scored on preserving identifying features. By comparing the verification rate, a method can be scored on the similarity of the faces. 3D facial recognition was able to give aditional insight in results which would otherwise be very similar and can be considered a good performance measurement for generating 3D facial data.

By transforming 3D to a 2D matrix it was possible to train a CWGAN to produce 3D data which was able to be correctly identified in 63.3% of the cases. Though improvement is needed for real world purposes, this shows that there is viability in this method.

A. Future Work

From the research a few recommendations for future work arose.

1) Bigger dataset: As discussed in multiple sections of this research the size of the dataset is too small for effective training. By combining FRGC v2.0, Bosphorus and Casia 3D, a dataset of 13,297 samples could be constructed, this would probably increase the performance of this method considerably. Additionally other methods of augmenting could be considered, such as flipping the faces on the x axis.

2) Asymmetric Euclidean loss: As noted by Tran et al. [22], deep learning on 3DMM had the tendency to return faces close to the origin of the 3DMM. Though our method does not use 3DMM, forcing the loss to focus on unique and identifying traits aligns with the goals of this research. Designing a GAN which uses asymmetric Eclidean loss can be beneficial for generating identifiable 3D data.

3) Incremental learning: Research has shown that training on smaller data first and then extending the model to increase the data size performs better compared to directly training the larger model [51]. As both the 2D and 3D information can easily be converted to lower resolution versions this could be beneficial for this method. Additionally one could look into different sizes for the 2D and 3D information as this research only considered equal size of these.

REFERENCES

[1] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3d data: A survey," *ACM Comput. Surv.*, vol. 50, pp. 20:1–20:38, Apr. 2017.

[2] B. K. Horn, "Shape from shading: A method for obtaining the shape of a smooth opaque object from one view," 1970.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[4] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): A survey," *IEEE Access*, vol. 7, pp. 36322–36333, 2019.

[5] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[6] C. Li, M. Zaheer, Y. Zhang, B. Póczos, and R. Salakhutdinov, "Point cloud GAN," *CoRR*, vol. abs/1810.05795, 2018.

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[8] D. Jiang, Y. Hu, S. Yan, L. Zhang, H. Zhang, and W. Gao, "Efficient 3d reconstruction for face recognition," *Pattern Recognition*, vol. 38, no. 6, pp. 787 – 798, 2005. Image Understanding for Photographs.

[9] S. Yan, M. Li, H. Zhang, and Q. Cheng, "Ranking prior likelihood distributions for bayesian shape localization framework.," in *ICCV*, vol. 1, pp. 51–58, 2003.

[10] S. Romdhani and T. Vetter, "Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 986–993 vol. 2, June 2005.

[11] Romdhani and Vetter, "Efficient, robust and accurate fitting of a 3d morphable model," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 59–66 vol.1, Oct 2003.

[12] J. Roth, Y. Tong, and X. Liu, "Adaptive 3d face reconstruction from unconstrained photo collections," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4197–4206, June 2016.

[13] D. Zeng, Q. Zhao, S. Long, and J. Li, "Examplar coherent 3d face reconstruction from forensic mugshot database," *Image and Vision Computing*, vol. 58, pp. 193 – 203, 2017.

[14] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 296–301, Sep. 2009.

[15] C. Ferrari, G. Lisanti, S. Berretti, and A. D. Bimbo, "A dictionary learning-based 3d morphable shape model," *IEEE Transactions on Multimedia*, vol. 19, pp. 2666–2679, Dec 2017.

[16] J. Booth, E. Antonakos, S. Ploumpis, G. Trigeorgis, Y. Panagakis, and S. Zafeiriou, "3d face morphable models "in-the-wild"," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5464–5473, July 2017.

[17] V. Blanz, T. Vetter, *et al.*, "A morphable model for the synthesis of 3d faces.," in *Siggraph*, vol. 99, pp. 187–194, 1999.

[18] V. Blanz and T. Vetter, "Face recognition based on fitting a 3d morphable model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 1063–1074, Sep. 2003.

[19] R. van Rootseler, L. Spreeuwers, and R. Veldhuis, "Application of 3d morphable models to faces in video images," in *32nd WIC Symposium on Information Theory in the Benelux* (O. van den Biggelaar, ed.), (Netherlands), pp. 34–41, Werkgemeenschap voor Informatie- en Communicatietheorie (WIC), 5 2011. http://opera.ulb.ac.be/wicsp2011/free/WICSP2011_Proceedings.pdf.

[20] M. D. Levine and Y. C. Yu, "State-of-the-art of 3d facial reconstruction methods for face recognition based on a single 2d training image per person," *Pattern Recognition Letters*, vol. 30, no. 10, pp. 908 – 913, 2009.

[21] D. Keith, "A Brief History of Deep Learning," *DATAVERSITY*, p. 07, 2 2017.

[22] A. T. Tran, T. Hassner, I. Masi, and G. G. Medioni, "Regressing robust and discriminative 3d morphable models with a very deep neural network," *CoRR*, vol. abs/1612.04904, 2016.

[23] A. Jourabloo and X. Liu, "Large-pose face alignment via cnn-based dense 3d model fitting," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4188–4196, June 2016.

[24] E. Richardson, M. Sela, and R. Kimmel, "3d face reconstruction by learning from synthetic data," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 460–469, Oct 2016.

[25] E. Richardson, M. Sela, R. Or-El, and R. Kimmel, "Learning detailed face reconstruction from a single image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5553–5562, July 2017.

[26] P. Dou, S. K. Shah, and I. A. Kakadiaris, "End-to-end 3d face reconstruction with deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1503–1512, July 2017.

[27] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos, "Large pose 3d face reconstruction from a single image via direct volumetric cnn regression," *International Conference on Computer Vision*, 2017.

[28] S. Sengupta, A. Kanazawa, C. D. Castillo, and D. W. Jacobs, "Sfsnet : Learning shape, reflectance and illuminance of faces in the wild," *CoRR*, vol. abs/1712.01261, 2017.

[29] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Pérez, and C. Theobalt, "Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," *CoRR*, vol. abs/1703.10580, Oct 2017.

[30] L. Tran and X. Liu, "Nonlinear 3d face morphable model," *CoRR*, vol. abs/1804.03786, 2018.

[31] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li, "Photorealistic facial texture inference using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5144–5153, 2017.

[32] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How generative adversarial networks and their variants work: An overview," *ACM Comput. Surv.*, vol. 52, pp. 10:1–10:43, Feb. 2019.

[33] Y. Cao, L. Jia, Y. Chen, N. Lin, C. Yang, B. Zhang, Z. Liu, X. Li, and H. Dai, "Recent advances of generative adversarial networks in computer vision," *IEEE Access*, vol. 7, pp. 14985–15006, 2019.

[34] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, pp. 141–142, Nov 2012.

[35] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017.

[36] L. Galteri, C. Ferrari, G. Lisanti, S. Berretti, and A. D. Bimbo, "Deep 3d morphable model refinement via progressive growing of conditional

generative adversarial networks," *Computer Vision and Image Understanding*, 2019.
[37] S. Z. Gilani, A. Mian, and P. Eastwood, "Deep, dense and accurate 3d face correspondence for generating population specific deformable models," *Pattern Recognition*, vol. 69, pp. 238 – 250, 2017.
[38] S. Moschoglou, S. Ploumpis, M. Nicolaou, A. Papaioannou, and S. Zafeiriou, "3dfacegan: Adversarial nets for 3d face representation, generation, and translation," *CoRR*, vol. abs/1905.00307, 2019.
[39] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou, "Joint 3d face reconstruction and dense alignment with position map regression network," *CoRR*, vol. abs/1803.07835, 2018.
[40] F. U. Nuha and Afiahayati, "Training dataset reduction on generative adversarial network," *Procedia Computer Science*, vol. 144, pp. 133 – 139, 2018. INNS Conference on Big Data and Deep Learning.
[41] L. Galteri, C. Ferrari, G. Lisanti, S. Berretti, and A. Del Bimbo, "Coarse-to-fine 3d face reconstruction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.
[42] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, Jin Chang, K. Hoffman, J. Marques, Jaesik Min, and W. Worek, "Overview of the face recognition grand challenge," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 947–954 vol. 1, June 2005.
[43] A. Savran, N. Alyüz, H. Dibeklioğlu, O. Çeliktutan, B. Gökberk, B. Sankur, and L. Akarun, "Bosphorus database for 3d face analysis," in *Biometrics and Identity Management* (B. Schouten, N. C. Juul, A. Drygajlo, and M. Tistarelli, eds.), (Berlin, Heidelberg), pp. 47–56, Springer Berlin Heidelberg, 2008.
[44] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.
[45] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," *CoRR*, vol. abs/1604.07379, 2016.
[46] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, "ESRGAN: enhanced super-resolution generative adversarial networks," *CoRR*, vol. abs/1809.00219, 2018.
[47] L. Spreeuwers, "Fast and accurate 3d face recognition," *International Journal of Computer Vision*, vol. 93, pp. 389–414, Jul 2011.
[48] L. Spreeuwers, "Setting a world record in 3d face recognition," *Vonk*, vol. 33, pp. 11–21, 11 2015.
[49] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.
[50] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.
[51] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *CoRR*, vol. abs/1710.10196, 2017.

APPENDIX A
WGAN

A. WGAN Generator

The generator has an latent vector of size 100 as input. Grayscale is assumed here otherwise the last output size would have been three.

Listing 1. WGAN generator

```
Conv1( in = 100, out = 1024, kernel size = 4, stride = 1, padding = 0 )
BatchNorm( features = 1024 )
Relu
Conv2( in = 1024, out = 512, kernel size = 4, stride = 2, padding = 1 )
BatchNorm( features = 512 )
Relu
Conv2( in = 512, out = 256, kernel size = 4, stride = 2, padding = 1 )
BatchNorm( features = 256 )
Relu
Conv3( in = 256, out = 128, kernel size = 4, stride = 2, padding = 1 )
BatchNorm( features = 128 )
Relu
Conv5( in = 128, out = 1, kernel size = 4, stride = 2, padding = 1 )
Tanh
```

B. WGAN Discriminator

The discriminator has the full image as input (1x64x64). Again grayscale is assumed.

Listing 2. WGAN discriminator

```
Conv1( in = 1, out = 128, kernel size = 4, stride = 2, padding = 1 )
BatchNorm( features = 128 )
LeakyRelu( negative_slope = 0.2 )
Conv2( in = 128, out = 256, kernel size = 4, stride = 2, padding = 1 )
BatchNorm( features = 256 )
LeakyRelu( negative_slope = 0.2 )
Conv3( in = 256, out = 512, kernel size = 4, stride = 2, padding = 1 )
BatchNorm( features = 512 )
LeakyRelu( negative_slope = 0.2 )
Conv4( in = 512, out = 1024, kernel size = 4, stride = 2, padding = 1 )
BatchNorm( features = 1024 )
LeakyRelu( negative_slope = 0.2 )
Conv5( in = 1024, out = 1, kernel size = 4, stride = 1, padding = 0 )
```

APPENDIX B
CWGAN

The CWGAN Generator contains an optional noise variable, a generator and a discriminator.

A. CWGAN Generator

The Generator consists of an Encoder and a Decoder.

1) Encoder: The input has twice the number of channels as the image as the added noise is equal in size to the image.

Listing 3. CWGAN generator encoder

```
Conv2d( in = im channels * 2, out = 128, kernel size = 4, stride = 2, padding = 1 )
InstanceNorm2d( 128, affine = True )
LeakyReLU( 0.2, inplace = True )
Conv2d( in=128, out = 256, kernel size = 4, stride = 2, padding = 1 )
InstanceNorm2d( 256, affine = True )
LeakyReLU( 0.2, inplace = True)
Conv2d( in = 256, out = 512, kernel size = 4, stride = 2, padding = 1)
InstanceNorm2d( 512, affine = True )
LeakyReLU( 0.2, inplace = True )
Conv2d( in = 512, out = 1024, kernel size = 4, stride = 2, padding = 1 )
InstanceNorm2d( 1024, affine = True )
LeakyReLU( 0.2, inplace = True )
```

2) Decoder: The output has the same number of channels as the image as the added noise is equal in size to the image.

Listing 4. CWGAN generator decoder

```
ConvTranspose2d( in = 1024, out = 512, kernel size = 4, stride = 2, padding = 1 )
BatchNorm2d( features = 512 )
ReLU( True )
ConvTranspose2d(in = 512, out = 256, kernel size = 4, stride = 2, padding = 1 )
BatchNorm2d( features = 256 )
ReLU( True )
ConvTranspose2d( in = 256, out = 128, kernel size = 4, stride = 2, padding = 1 )
BatchNorm2d( features = 128 )
ReLU( True )
ConvTranspose2d( in = 128, out =im channels, kernel size = 4, stride = 2, padding = 1 )
Tanh()
```

B. CWGAN Discriminator

Since the output is no longer a probability, the sigmoid is not applied.

Listing 5. CWGAN discriminator

```
Conv2d( in = 2 * im channels, out = 128, kernel size = 4, stride = 2, padding = 1 )
InstanceNorm2d( 128, affine = True )
LeakyReLU( 0.2, inplace = True )
Conv2d( in = 128, out = 256, kernel size = 4, stride = 2, padding = 1 )
InstanceNorm2d( 256, affine = True )
LeakyReLU( 0.2, inplace = True )
Conv2d( in = 256, out = 512, kernel size = 4, stride = 2, padding = 1 )
InstanceNorm2d( 512, affine = True )
LeakyReLU( 0.2, inplace = True )
Conv2d( in =512, out =1024, kernel size = 4, stride = 2, padding = 1 )
InstanceNorm2d( 1024, affine = True )
LeakyReLU( 0.2 , inplace = True )
Conv2d( in_channels = 1024, out = 1, kernel size = 4, stride = 1, padding = 0 )
```