Synthesizing Facial Photometries and Corresponding Geometries Using Generative Adversarial Networks

GIL SHAMAI, RON SLOSSBERG, and RON KIMMEL, Technion–Israel Institute of Technology, Israel

Artificial data synthesis is currently a well-studied topic with useful applications in data science, computer vision, graphics, and many other fields. Generating realistic data is especially challenging, since human perception is highly sensitive to non-realistic appearance. In recent times, new levels of realism have been achieved by advances in GAN training procedures and architectures. These successful models, however, are tuned mostly for use with regularly sampled data such as images, audio, and video. Despite the successful application of the architecture on these types of media, applying the same tools to geometric data poses a far greater challenge. The study of geometric deep learning is still a debated issue within the academic community, as the lack of intrinsic parametrization inherent to geometric objects prohibits the direct use of convolutional filters, a main building block of today’s machine learning systems.

In this article, we propose a new method for generating realistic human facial geometries coupled with overlaid textures. We circumvent the parametrization issue by utilizing a specialized non-rigid alignment procedure, and imposing a global mapping from our data to the unit rectangle. This mapping enables the representation of our geometric data as regularly sampled 2D images. We further discuss how to design such a mapping to control the distortion and conserve area within the target image. By representing geometric textures and geometries as images, we are able to use advanced GAN methodologies to generate new plausible textures and geometries. We address the often-neglected topic of relationship between texture and geometry and propose different methods for fitting generated geometries to generated textures. In addition, we widen the scope of our discussion and offer a new method for training GAN models on partially corrupted data. Finally, we provide empirical evidence demonstrating our generative model’s ability to produce examples of new facial identities, independent from the training data, while maintaining a high level of realism—two traits that are often at odds.

CCS Concepts: • Computing methodologies → Computer vision representations; Mesh geometry models; Parametric curve and surface models; Shape analysis; Image processing;

Additional Key Words and Phrases: Generative adversarial networks, 3D morphable model, face recognition, 3D face synthesis, data augmentation, facial texture and geometry

G. Shamai and R. Slossberg contributed equally to the paper.

This research was partially supported by the Israel Ministry of Science, grant number 3-14719, and the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel Cyber Bureau.

Authors’ addresses: G. Shamai and R. Slossberg, 435, Taub Building, Department of Computer Science, Technion – Israel Institute of Technology, Haifa 3200003, Israel; emails: [sgils, ronslos]@campus.technion.ac.il; R. kimmel, 439, Taub Building, Department of Computer Science, Technion – Israel Institute of Technology, Haifa 3200003, Israel; email: ron@cs.technion.ac.il.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.
1551-6857/2019/10-ART87 $15.00
https://doi.org/10.1145/3337067

ACM Trans. Multimedia Comput. Commun. Appl., Vol. 15, No. 3s, Article 87. Publication date: October 2019.
1 INTRODUCTION

The generation of realistic examples of everyday objects is a challenging and interesting problem that relates to several research fields such as geometry, computer graphics, and computer vision. The ability to capture the essence of a class of objects is key to the task of generating diverse datasets, which may be used in turn during the training of many machine learning based algorithms. The main challenge posed by the task of data generation is to construct the model that is able to generalize to many variations, while still maintaining high detail and quality. Furthermore, the challenge of generating geometric data is even greater due to the lack of intrinsic parameterization, inhibiting the use of the convolution operator.

In this work, we propose to learn the latent space of 3D textured objects, and focus our efforts on human faces. We show that by using a canonical transformation that maps geometric data to images, we are able to learn the distribution of such images via the GAN framework. By representing both texture and geometry of the face as images, we can learn the underlying distribution of faces and, as a consequence, generate new faces at will.

The generation of realistic human faces is a useful tool with applications in face recognition, puppetry, reconstruction, and rendering. Our main contributions are the proposition of a new model for 3D human faces that is composed in the 2D image domain, as well as the modeling of the relation between texture and geometry, further improving realism. By generating geometries and textures using state-of-the-art GANs, it is possible to create highly detailed data samples while maintaining the ability to generalize to unseen data—two, often conflicting, desirable properties.

While deep learning and convolutional networks have revolutionized many fields in recent years, they have been mostly employed on structured data that is intrinsically ordered. Arranged data such as audio, video, images, and text can be processed according to the order of samples, frames, pixels, or words. This inherent ordering permits the application of convolution operations that are the main building block of convolutional networks, a powerful and popular variant of deep networks. Contrary to typical parameterized data, geometric data, represented by two-dimensional manifolds, lacks an intrinsic parameterization, and is therefore more difficult to process via convolutional networks. This important class of data is crucial to the task of modeling our world, as most solid objects can be represented by a closed manifold accompanied by a texture overlay.

Recently, geometric data has grown in availability as more accurate and affordable acquisition devices have come into use. This abundance of data has attracted the attention of the computer vision and machine learning communities, which in turn has led to many new approaches for modeling and processing of geometries. One family of techniques for geometric data processing aims to define new operators that can be applied directly to the manifold and are able to replace to some extent the convolution operation within the processing pipeline. Other methods attempt to process geometries in the spectral domain or represent them in voxel space. These families of methods each have their merits but suffer from other issues such as loss of generality and memory inefficiency. In contrast, we propose to transform our geometric data via a canonical mapping into two-dimensional gridded data. This allows us to process the geometric data as images. The main advantage of this scheme is the ability to harness the abundance of fine-tuned CNN architectures available today.
The main contributions of this article are the following:

1. We propose a new non-linear model for generation of realistic textured facial geometries.
2. We propose a universal parametrization that weighs different features in the face according to their importance.
3. We propose a novel GAN model for exploiting incomplete or corrupted data.
4. We perform novel experiments, demonstrating our claims regarding the distribution of synthesized identities.

The article is organized as follows: In Section 2, we overview related works. In Section 3, we briefly review the construction of the 3D morphable model and its utilization for 3D face synthesis. In Section 4, we review the recent work titled \textit{progressive growing of GAN’s} used as part of our model. In Section 5, we present in detail our proposed data formation pipeline. In Section 6, we propose a new GAN model for exploiting corrupted data. In Section 7, we present our method for generating new textures and geometries, and discuss several proposals for providing matching geometries to given textures. Section 8 is devoted to the introduction of geometric expressions to our generated faces. Finally, Section 9 provides support for the proposed method by presenting quantitative as well as qualitative results.

2 RELATED WORK

Data augmentation is a common practice within the machine learning community. By applying various transformations to existing data samples it is possible to simulate a much larger data-set than is available and introduce robustness to transformations. A more advanced method for data augmentation takes into account the geometry of the scene. The technique that we term \textit{geometric data augmentation} consists of a geometry recovery stage, then transformation is performed on the geometry and finally a new image is created by projecting the geometry. In Reference [21], the authors show that by performing geometric data augmentation on a data-set of facial images they are able to reach state-of-the-art results on difficult facial recognition benchmarks. Despite its proven usefulness, geometric augmentation still lacks the ability to create completely new data samples outside the scope of the data-set.

A complementary method to data augmentation is data generation. By constructing a high-quality model for data generation it is possible to produce an infinitely large data-set. In addition, some models may permit control over the characteristics of each data sample. Within the domain of faces this would mean control over parameters such as age, gender, expression, pose, and lighting conditions. When dealing with image data a recent popular approach is to use a GAN [13], which is in essence a neural network with a trainable loss function. While this class of methods is well suited for images, reformulation in the context of geometry is more challenging and several competing approaches exist in this field. References [12] and [30] propose to construct samples from a low quality linear model, and then use a GAN to enforce the realism of the data. References [19] and [23] both propose the use of convolutional autoencoders that are trained on pre-aligned geometrical data. These methods, however, do not take into account the model texture. In addition, Reference [37] has used the popular voxel grid representation for geometries, and are able to generate 3D objects using this notion. This method, however, is memory inefficient and in practice can produce only coarse geometries.

In addition to data augmentation and generation, the objective of pose normalization is to decouple the subjects identity from other factors such as expression and pose, which may confuse a classifier. This can be either done by geometric reconstruction manipulation of the facial geometry or by performing normalization directly in the image domain. While References [9] and [2] leverage a geometric representation to transform the data, References [35] and [15] are able to
frontalize faces directly in the image domain as part of their pipeline. Although useful methods help the training process by limiting data variation, these methods still do not explicitly model new data samples, which is our ultimate goal.

An additional method for geometrically manipulating facial data that has gained success is geometric reconstruction from a single image. One popular family of methods aim to fit a parametric model to an image. This idea was first introduced by Reference [3] and has since been extended by works such as Reference [5]. An approach that involves regressing the coefficients of a given model via a deep network was first suggested by Reference [24] and extended by References [25] and [33]. More recently, methods that are not restricted to a specific model or attempt to learn the model during training time, such as References [28], [32], and [34], have been able leave the restricting assumptions of linear models such as 3DMM. Complementary efforts such as Reference [10] propose to reconstruct occluded texture regions to gain a full textured reconstruction from challenging poses as well. Another recent paper by Reference [27] focuses on improving the quality of facial texture used in reconstructed faces to improve realism. An additional complementary approach proposed by Reference [26] is to learn a direct mapping from an image to a template model. All of the above approaches, while useful, are based on fitting some geometry to a given image by relying on some underlying geometric model. This model, however, is not explicitly used to generate novel faces but rather to reconstruct existing ones.

Our most direct competition comes from several works in the field of facial generative modeling. The seminal work by Reference [3], which pioneered the field almost two decades ago, is still widely used within many methods, some of which were mentioned above. The linear 3D Morphable Model proposed is extremely flexible; however, it has the drawback of using a small number of PCA vectors that limit its ability to present highly detailed models. A recent large scale effort taken by References [7] and [6] has produced the largest publicly known 3DMM by scanning 10K subjects and using their scans to construct the model. In contrast to linear models, much more complex relations can be captured by training deep networks to take the part of data generators. To this end, References [32] and [34] were able to jointly learn a reconstruction encoder while also learning the facial model itself. Given the trained model, one could plausibly generate faces, however, the authors have not shown any experiments to this effect. Reference [23], however, has employed mesh autoencoders to construct new facial geometries; but this method does not produce texture and was trained on a limited data-set of very few subjects. In this article, we propose a new GAN based facial geometric generative model, and analyze the ability of our model to extend to new identities. We also relate between the geometric and texture models that are intrinsically correlated and discuss different ways of exploiting this correlation for our cause. While previous geometric models are used in the context of various tasks such as reconstruction or data compression, our model is, to the best of our knowledge, the first since 3DMM to propose a method for facial synthesis.

2.1 3D Morphable Model

One of the early attempts to capture facial geometry and photometry (texture) by a linear low dimensional space is the Blanz and Vetter [3] 3D Morphable Model (3DMM). Using the 3DMM, textures and geometries of faces can be synthesized as a linear combination of an orthogonal basis. The basis is constructed from a collection of aligned facial scans by applying the principal component analysis. Hence, the basis construction process relies on a vertex-to-vertex alignment of the facial scans, which is achieved by computationally finding a dense correspondence between each scan to a template model. The aligned vertices provide a set of spatial and texture coordinates that are then decomposed into the principal components of the set. Once the basis is constructed, it
is possible to represent each face by projecting it onto the first $k$ components of both the geometry and the texture bases.

This linear model was used to reconstruct 3D faces from 2D images; Blanz and Vetter [3] took an analysis-by-synthesis approach, which attempts to fit a projected surface model embedded in $\mathbb{R}^3$ into a given 2D image. This was carried out by constructing a fully differentiable parametric image formation pipeline, and performing a gradient descent procedure optimizing for an image to image loss on the model parameters. The parameters consist of the geometry and texture model’s coefficients of the face, as well as the lighting and pose parameters. This process results in a set of coefficients that encode the geometry and texture of any given face up to their projections on the principal components basis, effectively reconstructing the curved surface structure and the photometry of the given image of a face.

Even though two decades have passed since the inception of the 3DMM, it is still widely used in cutting-edge applications. By harnessing the generative powers of this model, it has been used as a tool for data augmentation and data generation for training of convolutional networks [12, 24, 25, 28]. Furthermore, the model has been integrated into deep learning pipelines to provide structure and regularization to the learning process [32].

3 3DMM CONSTRUCTION AND SYNTHESIS

According to the 3DMM model, each face is represented as an ordered set of $m$ geometric coordinates $g = (\hat{x}^1, \hat{y}^1, \hat{z}^1, \ldots, \hat{x}^m, \hat{z}^m) \in \mathbb{R}^{3m}$ and texture coordinates in RGB space $t = (\hat{r}^1, \hat{g}^1, \hat{b}^1, \ldots, \hat{r}^m, \hat{b}^m) \in \mathbb{R}^{3m}$. Given a set of $n$ faces, each represented by geometry $g_i$ and texture $t_i$ vectors, construct the $3m \times n$ matrices $G$ and $T$ by column-wise concatenation of all geometric coordinates and all corresponding texture coordinates. Since the alignment process ensures an ordered universal representation of all faces, Principal Component Analysis (PCA) [16] can be applied to extract the optimal first $k$ orthogonal basis components in terms of $L_2$ reconstruction error. To that end, denote by $V_g$ and $V_t$ the $3m \times n$ matrices that contain the left singular vectors of $\Delta G = G - \mu_g \mathbb{1}^T$ and $\Delta T = T - \mu_t \mathbb{1}^T$, respectively, where $\mu_g$ and $\mu_t$ are the average geometry and texture of the faces and $\mathbb{1}$ is a vector of ones.

By ordering $V_g$ and $V_t$ according to the magnitude of the singular values in a descending order, the texture and the geometric coordinates of each given face can be approximated by the linear combination

$$g_i \approx \mu_g + V_g \alpha_{g_i}, \quad t_i \approx \mu_t + V_t \alpha_{t_i}, \quad (1)$$

where $\alpha_{g_i}$ and $\alpha_{t_i}$ are the coefficients vectors, obtained by $\alpha_{g_i} = V_g^T (g_i - \mu_g)$ and $\alpha_{t_i} = V_t^T (t_i - \mu_t)$.

The 3D morphable model is useful not only in the context of representation and reconstruction, but it also allows for the generation of new faces that can not be found in the training-set. This can be done by randomly selecting the geometry and texture coefficients and plugging them into Equation (1). According to Reference [3], the distribution of the coefficients can be approximated as a multivariate normal distribution, such that the probability for a coefficient vector $\alpha$ is given by

$$P(\alpha) \sim \exp \left\{ \frac{-1}{2} \alpha^T \Sigma^{-1} \alpha \right\}, \quad (2)$$

where $\Sigma$ is a covariance matrix that can be empirically estimated from the data, and is generally assumed to be diagonal.

As is common practice when dealing with principal components, only the first $k \ll n$ vectors can be taken into account as part of the model. The number $k$ can be obtained by analyzing the decay of the singular values, which is proportional to the error produced by ignoring the associated basis vector. By excluding the vectors for which the singular variables are sufficiently small, we can guarantee minimal loss of data. In spite of the wide use and apparent success of the 3DMM it
is clear that the faces obtained from it tend to be over-smoothed and in most cases non-realistic. Furthermore, the multivariate normal distribution model from which the coefficients are drawn is over-simplified and does not represent the true distribution of faces. In particular, the texture and geometry are treated as two uncorrelated variables, contradicting empirical evidence. As a convincing visualization, we computed the canonical correlation between the 3DMM texture and geometry coefficients, \( \{ \alpha_{ti} \}_{i=1}^{n} \) and \( \{ \alpha_{gi} \}_{i=1}^{n} \), of the facial scans. Figure 1(a) shows \( u \) w.r.t. \( v \), the first two canonical variables of the correlation. Figure 1(b) shows a few samples of synthesized 3DMM faces and demonstrates the difference between the distributions of 3DMM generated faces and real ones.

4 PROGRESSIVE GROWING GAN

Generation of novel plausible data samples requires learning the underlying distribution of the data. Given a perfect discriminator that can differentiate between real and fake data samples, it is possible to construct a training loss for a generator model that tries to maximally confuse the discriminator. For complex realistic data, finding such a discriminator is a difficult problem on its own and requires learning from realistic and fake examples.

The fundamental idea of the GAN framework is to train both of these networks simultaneously. Essentially, this means that we use a trainable loss function for the generator that constantly evolves as the generator improves. This process can be formulated as Equation (3):

\[
\min_{G} \max_{D} \mathbb{V}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_{z}(z)}[\log(1 - D(G(z)))],
\]

where \( D, G \) are the discriminator and generator parametric functions, \( x, z \) are the real data samples and latent representation vectors, respectively.

Since we wish to produce high resolution textures for facial geometries, we propose to use a recent successful GAN, namely Reference [17]. The progressive growing GAN is constructed from layers, gradually increasing the resolution of the output image. During the training process, each layer is added consecutively while smoothly blending the new layers into the output as they are added. This, and several other techniques, were shown to increase the training stability as well as the variation of the generated data samples.

ACM Trans. Multimedia Comput. Commun. Appl., Vol. 15, No. 3s, Article 87. Publication date: October 2019.
The difficulty concerning geometric data is that it lacks the regular intrinsic ordering that exists in 2D images, which are essentially large matrices. For this reason, it is unclear how to apply spatial filtering, which is the core building block of convolutional network layers, to arbitrary geometric structures. Significant progress has been made in this direction by several recent papers. A comprehensive survey is presented in Reference [8]. These methods, however, are not yet widely used and supported within standard deep learning coding libraries. To harness the full power of recent state-of-the-art developments in the field, it is preferable to work in the domain of images. For this reason, we built a data processing pipeline that maps the geometric scanned data into a flat canonical image that allows the utilization of the progressively growing GAN without major modifications.

5 TRAINING DATA CONSTRUCTION

In this section, we describe the process by which we produce our training data. Our raw data consists of high quality geometric scans of human facial geometry and photometry, obtained from numerous subjects. By making use of a surface-to-surface alignment process [36], we are able to bring all the scans into correspondence with each other. Next, applying a universal mapping from the mesh to the 2D plane, we can transfer the facial texture into a canonically parametrized image. These aligned texture images are used to train our texture generation model. The training data construction process is depicted in Figure 2.

We provide several alternatives for constructing the facial geometry that accompanies each texture. One solution is to learn the relation between 3DMM texture and geometry coefficients, which is prevalent in the training data. In addition, we can similarly process the geometric data of the faces as well. By applying the same canonical transformation and encoding the \((x, y, z)\) coordinates of the model vertices as RGB channels of an image, we can learn to generate geometries as well as textures using the same methodology.

5.1 Face Scanning and Marking

Our training data formulation pipeline is designed for processing raw digital high resolution geometric scans of faces. Using a 3DMD scanner, roughly 1K different subjects were scanned, each making five distinct facial expressions including a neutral expression. The subjects were selected to form a balanced mixture of genders and ethnic backgrounds. Each scan is composed of the facial geometry, represented by a triangulated mesh, as well as two high resolution photographs that capture a 180-degree view of the subject’s face. Each mesh triangle is automatically mapped to one of the photos, allowing the facial texture to be transferred onto the mesh.

Due to the variety of facial geometries, as well as limitations of the scanning process, the meshes may contain imperfections such as holes and areas of missing texture. These data corruptions may
affect the training data samples that are presented to the network and care must be taken not to hinder the training process. The straightforward path to avoid corrupting the network with such data is to filter out the erroneous data samples completely. This leads to a significant reduction in the overall size of the training set size of roughly 20%. Instead, we propose a new approach that incorporates corrupted scans without compromising the integrity of the training data. We describe our approach to learning from corrupted data in Section 6.

To facilitate the alignment process described in Section 5.2, we annotate each face by 43 landmark locations. These locations are determined automatically by projecting the facial surface into an image and applying one of many 2D facial landmark detectors such as Dlib [18]. The landmarks are then back-projected onto the surface to determine their location. Finally, the locations of the automatically generated landmarks are manually refined to prevent displacements that might lead to errors during the alignment process.

### 5.2 Non-rigid Alignment

The goal of the alignment process is to obtain a dense correspondence between a predefined template facial mesh and each one of the facial scans. Guided by the pre-computed landmarks, this process is performed by deforming the template’s vertices into each one of the scanned meshes. Initially, a rigid alignment between the scanned mesh and the template mesh is computed as the optimal rotation, translation, and uniform scaling between the mesh and template landmarks. The deformation process is then performed by defining a fitting energy, described by Reference [3], composed of three energy terms measuring both the correspondence and the smoothness of the template’s deformation.

The first term accumulates the distances between the facial landmark points on the template mesh and their corresponding landmarks on the scanned mesh. It is formulated as

$$E_{\text{ref}} = \sum_{i} \| \tilde{v}_{l_i} - r_i \|^2,$$

where $\tilde{v}_{l_i}$ and $r_i$ hold the 3D coordinates of landmark $i$ on the deformed template and the scan mesh, respectively. The second term accumulates the distances between the template points to their nearest scan points, as well as to their tangent planes. It is formulated as

$$E_{\text{fit}} = \sum_{i=1}^{n} w_i \left( |n_{c_i}^T (\tilde{v}_i - c_i) |^2 + 0.1 \| \tilde{v}_i - c_i \|^2 \right),$$

where $c_i$ is the nearest scan point to $\tilde{v}_i$, and $n_{c_i}$ is the normal of the tangent plane of $c_i$. $w_i$ is either 1 or 0 and can be used to discard template points that are, for example, unreliable, or too far from their nearest scan points. The third term serves as a regularization and penalizes non-smooth deformations, and is formulated as

$$E_{\text{memb}} = \sum_{i \in \mathcal{V}} \| \Delta d_i \|^2,$$

where $d_i$ is the deformation of vertex $\tilde{v}_i$ between the deformed and original template, and $\Delta$ is the Laplace-Beltrami operator that is defined on the original template mesh.

The template is deformed by minimizing the total energy

$$E_{\text{tot}} = E_{\text{fit}} + \alpha_{\text{ref}} E_{\text{ref}} + \alpha_{\text{memb}} E_{\text{memb}}$$

with respect to the deformed template vertices $\tilde{v}$, where $\alpha_{\text{ref}}, \alpha_{\text{memb}}$ are weights that can be adjusted. The non-rigid alignment process is done by iterating between deforming $\tilde{v}$ and finding the nearest neighbors $c$ within the scanned surface. As a result, a dense point-to-point correspondence is obtained between the template and the scanned meshes.
5.3 Universal Mapping

Given a scanned facial surface with unknown parametrization, our goal in this section is to discover a consistent 2D parameterization of the surface that maps it to a unit rectangle. We use this mapping to map each scan to the unit rectangle in a consistent manner. In Section 5.2, we described the process of aligning the facial surface template to a scanned facial surface bringing them into correspondence for this purpose. In the following section, we define the universal parameterization between the template face and the unit rectangle.

The authors of Reference [31] defined the mapping between the scan and the plane using a ray casting technique built into the animation rendering toolbox Blender [4]. Figure 3 depicts several examples of the resulting facial photometry maps. Although it would be possible to map via the same parametrization, we choose to introduce a new improved mapping scheme. The Blender mapping, for example, does not exploit the entire squared image for the mapping. Moreover, it does not take the facial structure into account. The eyes, nose, and mouth, for instance, clearly contain more details than smoother parts of the face such as the cheeks and forehead. It is reasonable to assume that it would be easier to learn and reconstruct the main features, perhaps at the expense of other parts if they take up a larger portion of the input images. To that end, we propose to construct a weighted parametrization that will allow us to control the relative area in the plane taken up by each facial feature.

In Reference [11], the authors present a parametrization technique that encodes each mapped vertex coordinate as a convex combination of the mapping of its first ring neighbors. The authors demonstrate that any set of barycentric coordinates has a unique planar graph with a valid triangulation that fulfills it. As an extension, the authors prove that any valid choice of barycentric coordinates provides a valid mapping that minimizes a weighted sum of the mapped edge lengths. The aforementioned method and its use within our work are briefly described below.

Given any triangulated mesh, the object is to map it into a valid planar graph with the same connectivity. Assuming a mesh with \( N \) vertices, choose a set of \( K \) boundary vertices from the mesh and fix their 2D mapping values to some desired convex boundary, \( u_1, \ldots, u_K \). For any other vertex \( i > K \) in the mesh, choose a set of non-negative barycentric coordinates \( \lambda_{i,j} \), such that \( \sum_{j=1}^{N} \lambda_{i,j} = 1 \), and \( \lambda_{i,j} = 0 \) if and only if \( i \) and \( j \) are not connected. Then, for \( i = K + 1, \ldots, N \), solve the linear system of equations

\[
    u_i = \sum_{j=1}^{N} \lambda_{i,j} u_j.
\]

The authors of Reference [11] prove that Equation (8) has a closed form unique solution that coincides with the chosen barycentric coordinates. According to Reference [11], it can be shown...
that for any desired set of weights defined as

$$\lambda_{i,j} = \frac{w_{i,j}}{\sum_{j:(i,j) \in E} w_{i,j}},$$

(9)

the solution of Equation (8) minimizes the functional $\sum_{j:(i,j) \in E} w_{i,j} \|u_i - u_j\|^2$, where $E$ represents the set of edges.

Following this notion, we designed the weights $w_{i,j}$ such that eyes, nose, and mouth would receive a larger relative area in the parametrization plane. We defined a weight for each vertex in the template face, and then gave each edge the average weight of its two adjacent vertices. Note that the resulting edge lengths also depend on the density of vertices in the mesh. In other words, when choosing a constant weight for all edges, the edge lengths of the resulting parametrization termed the uniform barycentric parametrization, is not constant. To design the edge weights more intuitively, we normalize the edge weights by the ones resulting from the uniform barycentric parametrization. A visualization of the edge weights is shown in Figure 3.

To choose the boundary vertices $u_1, \ldots, u_K$, we follow the outer boundary of the facial mesh, starting from the center bottom (a point on the chin), while measuring the length of edges we pass through, $L_1, \ldots, L_K$. Assume the image boundary is parametrized by $C(t) = \{x(t), y(t)\}$ for $0 \leq t \leq 1$, such that $C(0) = C(1)$ is the bottom center of the image. Then, we set $u_i = C(t_i)$, where

$$t_i = \frac{\sum_{j=1}^{i} L_j}{\sum_{j=1}^{K} L_j}.$$  

(10)

Last, unlike Reference [31], we propose to construct a symmetric mapping to augment the data by mirroring the training samples. To this end, we ensure that the template is intrinsically symmetric and take care to map the boundary vertices symmetrically to the mapping boundary. The resulting mapping as well as a visualization of the edge weights are shown in Figure 3. The rightmost depiction in Figure 3 shows that when mapping back the unwrapped texture to the facial geometry, a better resolution is obtained in the eyebrow region under the proposed method.

6 LEARNING FROM CORRUPTED DATA

The semi-automatic data acquisition pipeline described in Section 5 is used to construct a data-set of 2D images used to train the GAN. Naturally, some of the generated data samples contain corrupted parts errors introduced during one or more of the pipeline stages. During the 3D scanning process, for example, facial textures that contain hair are often not captured correctly. Another reason for incomplete texture are occlusions and limited camera field of view. The geometry of the eyes is occasionally distorted due to their high specular reflectivity. The automatic landmark annotation stage may result in some inaccurate or erroneous landmarks, thus causing various distortions of the final output. Figure 4 depicts several examples of such data corruptions. One way
Fig. 5. The proposed GAN model for learning from incomplete data. A valid mask is obtained for each training sample according to the corrupted regions. The real and generated samples are concatenated and masked by the same valid masks and then presented to the discriminator network. The discriminator cannot distinguish between the real and fake images by their masks alone. The generator cannot generate corrupted images, since corruptions are masked. The generator does not have any information about the mask and therefore cannot generate unwanted black holes, since they would not fit the masked regions.

to handle data corruption is to ignore imperfect images and keep only the valid ones. In our case, manual screening of the data reduces the number of samples from a total of 4,963 samples to only 3,679 valid ones, thus, eliminating 25% of the data. Here, we propose a novel technique for training GANs using partially incomplete data able to exploit undamaged parts and robustly deal with corrupted data.

To that end, we propose to pair a binary valid mask to each training sample, marking the corrupted regions. Without loss of generality, black areas in the masks (zero values) correspond to corrupted regions we would like to ignore, and white regions (values of one) correspond to valid parts we would like to exploit during training. We propose to multiply these valid masks by their corresponding images, as well as concatenate them as a forth channel (R-G-B-mask). Recall that the discriminator receives as an input a batch of real images and a batch of generated fake images. To prevent the discriminator from distinguishing between real and fake images according to the valid mask, the exact same mask is multiplied and concatenated to both real and fake samples. The generator, which does not get the masks as an input, must produce complete images filling in the masked regions with valid pixels. Otherwise, the discriminator would be able to easily identify masked parts that do not match the valid masks and conclude that the image is fake. The valid masks could be constructed either manually or by using automatic image processing techniques. The discriminator and generator architectures of the proposed model are depicted in Figure 5.

To demonstrate the performance of the proposed GAN, we constructed a synthetic data-set of different colored shapes randomly located in 10K images of size $256 \times 256$. In this simple experiment, we treat the red circles as corruptions that we would like our model to ignore. Figure 6 shows the data images, the valid masks, and the resulting GAN output. It is clearly seen that the proposed GAN model generated new data images without the unwanted red circles. Furthermore, it is evident that the model is able to produce black circles that are part of the valid data. This implies that the model does not simply learn to ignore black regions.
Fig. 6. Left three: examples of the constructed synthetic data consisting of images containing shapes of various colors. In this experiment, red circles are unwanted in the output. Middle three: valid masks corresponding to the unwanted elements. Right three: examples of data samples obtained by the proposed GAN.

Fig. 7. Left: Facial textures generated by the suggested pipeline. Right: Real textures from the training-set.

7 FACIAL SURFACE GENERATOR

We propose to train a model capable of generating realistic geometries and photometries (textures or colors) of human faces. The training input to our model is constructed according to Section 5, and used to train a NN coupled with a GAN loss. At inference, the trained model is used to produce random plausible facial textures that are mapped according to our predefined parametrization described in Section 5.3. To also generate corresponding facial geometries for each new texture, we propose two novel approaches. The first approach is based on training a similar model for geometries. This is done by mapping the training-set geometric coordinates using the canonical parametrization into the unit rectangle. By treating each mapped coordinate as a color channel, we obtain geometry images used to train the geometry generator model. The second approach relies on the classical 3DMM model. For both approaches, we suggest a method to generate a plausible geometry for a given texture. In the following sections, we describe the two proposed approaches in detail.

7.1 Generating Textures Using GANs

Our texture generation model is based on a Convolutional Neural Network that is trained to minimize a GAN loss. By training our model on our data-set constructed according to Section 5, we are able to generate new plausible textures mapped to the unit rectangle according to the predefined parametrization described in Section 5.3. As we show in the following sections, the textures generated by the proposed model present novel yet realistic human faces. Since texture and geometry are both inseparable attributes of the same geometric entity, it is necessary to take the relationship between them into account when generating the corresponding geometries. In Section 7.3 and Section 7.2, we describe in detail the proposed geometry generation pipeline that takes as input a generated texture and produces a corresponding plausible geometry. Several examples obtained by the suggested texture generation model are depicted in Figure 7.

7.2 Assigning Geometries to Textures

Once novel textures have been generated, we would like to assign to them plausible synthetic geometries to obtain realistic face models. One way to generate geometries is by exploiting the 3DMM model by which geometries can be recovered through proper selection of the coefficients. In what follows, we discuss and compare several methods for obtaining the 3DMM geometry coefficients for the realistic synthesized geometries.
7.2.1 Random. The simplest way of synthesizing a geometry to a given texture is by picking random 3DMM geometry coefficients. We follow the formulation in Equation (2). The probability of a coefficient $\alpha_i$ is given by

$$P(\alpha_i) \sim \exp\left\{-\frac{\alpha_i^2}{2\sigma_i^2}\right\}, \quad (11)$$

where, $\sigma_i^2$ is the $i$th eigenvalue of the covariance of $\Delta G$. $\sigma_i^2$ can be computed more efficiently as $\sigma_i^2 = \frac{1}{n}\delta_i^2$, where $\delta_i$ is the $i$th singular value of $\Delta G$. To fit a geometry to a given texture, we randomize a vector of coefficients from the above probability distribution and reconstruct the geometry using the 3DMM formulation from Equation (1).

Random geometries are simple to generate; however, not every geometry can realistically fit any texture. Evidence of this is given in Figure 1(a) where the canonical correlation vectors between texture and geometry are plotted. In what follows, we proceed to generate geometries intended to suit a given generated texture.

7.2.2 Nearest Neighbor. Given a new texture, a simple way to fit a geometry that is likely to match it is by finding the data sample with the nearest texture, and projecting its geometry onto the 3DMM subspace. To this end, we define a distance between two textures as the $L_2$ norm between their 3DMM texture coefficients. Note that this setting requires only that the 3DMM texture and geometry coefficients of the data be stored. Nearest neighbor geometries are simple to obtain; however, they are restricted to the training data geometries alone.

7.2.3 Maximum a Posteriori. The maximum a posteriori (MAP) estimate is typically used when one can formulate assumptions about the data distribution. In our case, given input facial textures, MAP could be used to obtain the most likely geometries under a set of assumptions regarding the conditional distribution of the data. We first construct a mutual 3DMM basis by concatenating texture and geometry vectors.

We define a vertical concatenation of geometries $G$ and textures $T$ as the $6m \times n$ matrix $M = (G^T \ T)$, where $G, T$ were previously defined in Section 3. We define $\Delta M = M - \mu_M1^T$, where $\mu_M$ holds the average of the rows of $M$. Denote by $U$ the $6m \times k$ matrix that contains the first $k$ basis vectors of $\Delta M$, i.e., corresponding to the largest magnitude eigenvalues. These vectors can be computed either as eigenvectors of $\Delta M\Delta M^T$ or, more efficiently, as the left singular vectors of $\Delta M$. Denote by $U_g$ and $\mu_{M_g}$ the upper halves of $U$ and $\mu_M$, and denote by $U_t$ and $\mu_{M_t}$ the lower halves $U$ and $\mu_M$, respectively, such that

$$U = \begin{pmatrix} U_g \\ U_t \end{pmatrix}, \quad \mu_M = \begin{pmatrix} \mu_{M_g} \\ \mu_{M_t} \end{pmatrix}. \quad (12)$$

Note that $U_g$ and $U_t$, unlike $V_g$ and $V_t$ (which were defined in Section 3), are not orthogonal. Nevertheless, any geometry $g$ and texture $t$ of a given face in $M$ can be represented as a linear combination

$$\begin{pmatrix} g \\ t \end{pmatrix} = \begin{pmatrix} \mu_{M_g} \\ \mu_{M_t} \end{pmatrix} + \begin{pmatrix} U_g \\ U_t \end{pmatrix}\beta, \quad (13)$$

where the coefficients vector $\beta$ is mutual to the geometry and texture. Using the notations and definitions above, any new facial texture could be approximated through a coefficients vector $\beta$ as

$$t = U_t\beta + \mu_{M_t} + \text{noise}_t,$$

$$g = U_g\beta + \mu_{M_g} + \text{noise}_g. \quad (14)$$

Out MAP assumption is that $\beta, \text{noise}_t$, and $\text{noise}_g$ follow a multivariate normal distributions with zero mean. Given a facial texture $t$, our goal is to compute the most likely coefficient vector $\beta^*$.
under this assumption, and then obtain the most likely geometry as
\[ g = U_g \beta^* + \mu_{M_g}. \] (15)
Following Bayes’ rule, one could formulate the most likely coefficient vector as
\[ \beta^* = \arg\max_{\beta} P(\beta|t) \]
\[ = \arg\max_{\beta} \frac{P(t|\beta)P(\beta)}{P(t)} \]
\[ = \arg\max_{\beta} P(t|\beta)P(\beta). \] (16)
Since \( P(t|\beta) \) and \( P(\beta) \) follow multivariate normal distributions, denote their covariance matrices by \( \Sigma_{t|\beta} \) and \( \Sigma_{\beta} \), and their mean vectors by \( \mu_{t|\beta} = U_t \beta + \mu_{M_t} \) and \( \mu_{\beta} = \vec{0} \), respectively. Thus,
\[ \beta^* = \arg\max_{\beta} P(t|\beta)P(\beta) = \arg\max_{\beta} \exp \left\{ -\frac{1}{2} (t - \mu_{M_t} - U_t \beta)^T \Sigma_{t|\beta}^{-1} (t - \mu_{M_t} - U_t \beta) \right\} \]
\[ = \arg\min_{\beta} (t - \mu_{M_t} - U_t \beta)^T \Sigma_{t|\beta}^{-1} (t - \mu_{M_t} - U_t \beta) + \beta^T \Sigma_{\beta}^{-1} \beta. \] (17)
One could obtain a closed form solution for \( \beta^* \) by setting the derivative to zero to obtain
\[ \beta^* = \left( U_t^T \Sigma_{t|\beta}^{-1} U_t + \Sigma_{\beta}^{-1} \right)^{-1} U_t^T \Sigma_{t|\beta}^{-1} (t - \mu_{M_t}). \] (18)
We estimate the covariance matrices \( \Sigma_{\beta} \) and \( \Sigma_{t|\beta} \) empirically from the data. Since the mean of each coefficient in \( \beta \) with respect to all samples is zero, the covariance \( \Sigma_{\beta} \) can be estimated by
\[ \Sigma_{\beta} = \frac{1}{n-1} \sum_{i=1}^{n} \beta_i \beta_i^T, \] (19)
where, \( \beta_i \) is the coefficient vector for face sample \( i \). The \( 3m \times 3m \) covariance matrix \( \Sigma_{t|\beta} \) is very large, impractical to estimate from a few thousand of samples or to invert once estimated. Hence, for simplicity, we approximate it as a diagonal matrix that does not depend on \( \beta \). One can verify that the mean of each element in \( \text{noise}_i = U_t \beta + \mu_{M_t} - t \) with respect to all samples is zero. Hence, we estimate its \( j \)th diagonal value as
\[ \Sigma_{t|\beta, j} = \frac{1}{n-1} \sum_{i=1}^{n} \text{noise}_{i, j}^2. \] (20)

7.2.4 Least Squares. Least squares (LS) minimization is a simple yet powerful approach typically used when sufficient data samples are available. One can consider the use of such a method as training a multivariate linear regression with an \( L_2 \) loss. Assume a facial sample is represented by a texture vector \( t \) and a geometry vector \( g \). Denote by \( \alpha_t \) and \( \alpha_g \) the column vectors with the first \( k_t \) and \( k_g \) texture and geometry 3DMM coefficients of the face. These coefficients are obtained by projecting \( t \) and \( g \) onto the 3DMM basis \( V_t \) and \( V_g \). Let the \( k_t \times n \) matrix \( A_t \) hold the texture coefficient vectors of all samples in its columns, and let the \( k_g \times n \) matrix \( A_g \) hold the geometry coefficient vectors of all samples in its columns in the same order as \( A_t \). The correlation between \( A_t \) and \( A_g \) could be linearly approximated by
\[ A_g \approx W^T A_t. \] (21)
Fig. 8. Left: mapped textures from the test-set. Right: corresponding geometries reconstructed using each proposed approach. Average $L_2$ reconstruction error for each method appears above each method illustration.

Note that we would not benefit from generalizing from a linear to an affine correlation, since the mean of each row in $A_t$ and $A_g$ are zero, as they hold singular values of a centered set of samples. Following Equation (21), we wish to obtain $W$ as the minimizer of

$$\text{loss}(W) = \|W^TA_t - A_g\|_F.$$  

A closed form solution is easily obtained as

$$W^* = (A_tA_t^T)^{-1}A_tA_g^T = A_t^TA_g^T.$$

Defining $\tilde{V}_t$ and $\tilde{V}_g$ as holding the first $k_t$ and $k_g$ texture and geometry 3DMM basis vectors, we estimate $W^*$ using a set of training samples. Then, given a new texture $t$, one could fit a geometry $g$ by computing the texture coefficients as

$$\alpha_t = \tilde{V}_t^T(t - \mu_t),$$

computing the geometry coefficients as

$$\alpha_g = W^*\alpha_t,$$

and finally, computing the geometry as

$$g = \tilde{V}_g\alpha_g + \mu_g.$$  

7.2.5 Geometry Recovery Method Comparison. To evaluate our geometry recovery from texture, we use a test-set of textures and their corresponding geometries obtained from 297 scans excluded from both the GAN training and geometry fitting procedures. Given these unseen textures as input, we estimate their geometries using the approaches presented above, and compare them to their ground truth. We compare between the methods by computing the average $L_2$ norm between the vertices of the reconstructed and true geometries. For this experiment, we chose $k = k_t = k_g = 200$. Figure 8 shows examples of test-set textures mapped onto their assigned geometries, obtained by each of the aforementioned methods.

It is evident that the LS approach presents the lowest average reconstruction error across the test-set. Since it is also simple and efficient, we proceed to use the LS approach to approximate the geometries in the following sections. Note, however, that the alternative methods could be beneficial in other cases, depending on the specific application. Figure 9 visually compares between reconstructed geometries and the true ones for different textures from the test-set, using the LS approach. The $L_2$ norm between the reconstruction and true geometries are given below each example. The geometries, predicted solely from textures of identities that were never seen before, are surprisingly very similar to the real ones. This validates the strong correlation assumption between textures and geometries, as well as our ability to generate realistic geometries for unseen data.

7.3 Generating Geometries Using GANs

In Section 7.2, we presented our methods for geometry reconstruction based on the 3DMM model. We observe that projecting geometries onto the subspace of 3DMM has almost no visual effect on the appearance of the faces. The 3DMM, however, is constrained to the subspace of training-set
geometries and due to the simple linear model being incapable of producing examples outside the convex hull of the training set.

In Section 5.3, we mapped facial textures onto 2D images, with the goal of producing new textures. The same methodology can be used to produce new geometries as well. To that end, we propose to construct a data-set of aligned facial geometries and train a GAN to generate new ones by repeating the texture mapping process while replacing its RBG texture values by its XYZ coordinates.

As for data augmentation, while the amount of texture samples can be doubled by horizontal mirroring each of the images, we found that mirroring relative to each of the X, Y, and Z axes independently results in a valid facial geometry. Thus, the amount of geometry samples can be augmented by a factor of 8. Note, however, the mirroring relative to the X axis should be carried out as \( X_{\text{mirror}} = C - X \), where \( C \) is a constant that could be set, for example, to the maximal X value in all training samples.

To fit the GAN generated geometry to a given texture, one can repeat the geometry reconstruction methods in Section 7.2 using the latent representation vector from the GAN instead of the 3DMM coefficients. To apply the fitting methods, we would obtain the latent coefficients for each geometry and texture sample in the training-set. One way to accomplish this is by using gradient descent on the input latent space of the generator using an \( L_1 \) loss between the generator output and the desired training-set sample. We propose to initialize the latent coefficients for the gradient descent optimization to a reasonably close solution. This can be achieved by first using the generator to produce random images, for which the coefficient are known. Then, given a training image, initialize the latent coefficients to the ones of its nearest neighbor within the generated set. Once latent coefficients are obtained, we may use the same methodology described in Section 7.2 to produce latent geometry coefficients for a given texture. These coefficients when introduced to the geometry GAN model will produce a mapped geometry that we convert to a triangulated mesh according to the template connectivity. The training data and resulting generated geometries are shown in Figure 10.
Fig. 10. Left: the template aligned to four real 3D facial scans, colored by their X, Y, and Z values. Center-left: geometry values mapped to an image using the proposed universal mapping. These images are used to train a GAN. Center-right: fake geometry image generated by the GAN. Right: the synthesized geometries as triangulated meshes.

Fig. 11. Various expression vectors applied to a single face.

8 GEOMETRIC EXPRESSIONS

In Section 7, we propose a method for generating new facial textures and corresponding geometries. To further enrich our model, we propose to add geometric expressions to our generated faces as well. The unwrapped textures generated by the GAN include traces of the subject’s expressions within them. For example, the texture of a smiling face tends to have wrinkles in the cheeks. As a consequence, both geometry and expression were reconstructed during synthesis process. This can be seen in Figure 8 and Figure 9.

These reconstructed expressions, however, are limited to the expressions in the training data. Although this might result in a geometric expression that does not fit the textural expression, one would still possibly want to modify the geometric expression, for example, for data augmentation or graphic design purposes. To that end, we follow Reference [9], which defines a linear basis for expressions by taking the difference $g_{diff} = g_{expr} - g_{neutral}$ for every face in the set. We then remove the mean difference vector to obtain $\Delta G_{diff} = G_{diff} - \mu_{diff}$. We compute the principal components of $\Delta G_{diff}$ to obtain our geometric expression model. The expression difference model can be applied to a given face by randomizing the expression coefficients and adding the linear combination of the difference vectors to a generated neutral face as so: $g_{exp} = g_{neutral} + \mu_{diff} + \alpha g + V_{exp}$. Since the expression model must be applied to neutral faces, we define a model for the generation of expression neutral geometries. To span only the space of neutral expressionless faces, we suggest to replace all the geometries in our training-set with their neutral counterpart. By following this course, the texture model still benefits from all the textures available to us while our geometry model learns to predict only neutral models for any texture with or without expression. This method can be applied to either the 3DMM based geometry model from Section 7.2 or the GAN based geometry model described in Section 7.3. Figure 11 demonstrates the result of deforming the geometric expression of a single facial texture coupled with its fitted neutral geometry.

9 EXPERIMENTAL RESULTS

To demonstrate the ability of our model to generate new realistic identities, we performed several quantitative as well as qualitative experiments. We start by generating several random textures and obtain their corresponding geometries according to Section 7.2. We are able to vary the expression
Fig. 12. Several generated faces rendered with varying expression and pose under varying lighting conditions.

Fig. 13. Left: distribution of distances between training and generated IDs. Right: cumulative sum of distances between training and test in dark blue and generated to test in light blue.

by applying a linear expression model as described in Section 8. According to this model, each expression can be represented by a variation from the mean face, which leads to a specific facial movement. By combining various movements of the face, one can generate any expression desired. The faces are then rendered under various poses and lighting conditions. The rendered faces are depicted in Figure 12.

Our next qualitative experiment demonstrates the ability of our model to generate completely new textures by combining facial features from different training examples. To this end, we searched for the nearest neighbor to the generated texture from within the training data. It can be seen in Figure 13 that the demonstrated examples have nearest neighbors that are significantly different from them and cannot be considered as the same identity. Within the following section, we will analyze both the generative ability of our model to produce new faces as well as its realism and similarity to realistic examples. In addition, we also searched for generated texture samples that are nearest to several validation set examples. By finding close-by textures, we demonstrate the generalization capability of our model to unseen textures. This is demonstrated in Figure 14.
Fig. 14. Left: generated textures coupled with their nearest neighbor within the training-set. Right: validation textures coupled with the nearest out of 10K generated textures.

Table 1. Sliced Wasserstein Distance between Generated and Real Texture Images

| Resolution | 1,024 | 512  | 256  | 128  | 64   | 32   | 16   | avg  |
|------------|-------|------|------|------|------|------|------|------|
| Real       | 3.33  | 3.33 | 3.35 | 2.93 | 2.53 | 2.47 | 4.16 | 3.16 |
| **Proposed** | 35.32 | 18.13| 10.76| 6.41 | 7.42 | 10.76| 34.86| 17.67|
| PCA        | 92.7  | 224.01| 156.87| 66.20| 33.97| 104.08| 99.08| 99.08|

The previous qualitative assessment is complemented by a more in-depth examination of the nearest neighbors across 10K generated faces. For the following experiments, we had freedom to choose our distance metric. We aimed to find a natural metric that coincides with human perception of faces. We therefore chose to render each generated and real face from a frontal perspective and process the rendered images via a pre-trained facial recognition network. Using a model based on Reference [1], we extracted the final latent feature vector from within the network as our facial descriptor. The distance was calculated as $\text{dist} = ||D_1 - D_2||_2$ where $D_1, D_2$ are the descriptors corresponding to the first and second face, respectively. By analyzing the distribution of such distances, we can assess the spread of identities that exists within each data-set as well as the relation between different data-sets.

We used the distributions of distances between generated faces and the training and validation sets of real faces to assess the quality of our generative model. In Figure 13, we plot the distribution of distances between generated sample and their nearest real training sample. This plot demonstrates that the distance distribution resembles Gaussian distribution with non-zero mean. This implies that on average new identities are located in between the real samples and not directly near them. Our analysis of the distances to the neighbors of the validation set, also depicted in Figure 13, shows that our model is able to produce identities of subjects similar to ones found in the validation set that were not used to train the model. This validates our claim that our model produces new identities by combining features from the training-set faces, and that these identities can generalize to the unseen validation set.

Following Reference [17], we performed an analysis of sliced Wasserstein distance [22] on our generated textures and geometries. By assessing the distance between the distributions of patches taken from textures generated by our model relative to patches taken from faces generated by 3DMM, we analyzed the multi-resolution similarity between the generated and real examples. Table 1 and Table 2 show the SWD for each resolution relative to the training data. In both experiments it is clear that the SWD is lower for our model at every resolution, indicating that at every level of detail the patches produced by our model are more similar to patches in the training data.

In addition to assessing the patch level feature resemblance, we sought to uncover the distances between the distributions of identities. To this end, we conducted two more experiments that gauged the similarity between the distributions of generated identities to that of the real ones.
Table 2. Sliced Wasserstein Distance between Generated and Real Geometry Images

| Resolution | 1,024 | 512  | 256  | 128  | 64   | 32   | 16   | avg  |
|------------|-------|------|------|------|------|------|------|------|
| Real       | 6.08  | 2.41 | 3.40 | 2.45 | 3.10 | 2.75 | 1.86 | 3.15 |
| **Proposed** | 11.8  | 9.58 | 27.13| 44.5 | 38.05| 11.03| 2.16 | 20.61|
| PCA        | 272.7 | 43.94| 29.6 | 50.72| 44.43| 13.21| 4.67 | 65.61|

Fig. 15. T-SNE embedding of face IDs. Left: real versus generated IDs. Center: real versus 3DMM IDs. Right: generated IDs labeled according to real data clusters.

Table 3. Sliced Wasserstein Distance between Distributions of Identities from Different Sets

| Method | 3DMM | **Proposed** | train |
|--------|------|--------------|-------|
| train  | 59.88| 35.82        | -     |
| test   | 75.3 | 62.09        | 42.4  |

To qualitatively assess these distributions, we plot our identities using the common dimensionality reduction technique T-SNE [20]. Figure 15 depicts the low dimensional representation of the ID descriptors produced by our model and the 3DMM overlaid on top of the real data descriptor embedding. In addition, Figure 15 also depicts the clustering of different ethnic groups as well as gender as data points of different colors. By assigning each generated sample to the nearest cluster, we can automatically assign each new sample with its nearest cluster to obtain automatic annotation of our generated data. Furthermore, we performed a quantitative analysis of the difference between identity distribution using SWD. The results of this experiment are depicted in Table 3.

Last, we repeated the identity distribution and identity variation experiments to compare between the geometry reconstruction methods proposed in Section 7.2 and Section 7.3. To fit the geometries produced by the GAN to the textures, we follow the method proposed in Section 7.3, coupled with the LS method. We evaluated the reconstruction error by employing the method from Section 7.2.5, and summarize the reconstruction errors in the top row of Table 4. The computed error explicitly measures the ability of each method to faithfully reproduce each test-set geometry given an unseen texture. Next, we generated 3K random facial textures, and fitted their geometries using each one of the aforementioned methods, producing 3K textured 3D faces. We then rendered 2D frontal images from these faces and extracted the activations from the final layer of the recognition network as before. We repeated the identity variation experiment (as in Figure 13) and the SWD identity experiment (as in Table 3), for each one of the methods. The results of these experiments are summarized in Table 4. The NN to test column demonstrates the ability of each
method to create new identities and generalize beyond the training-set. The SWD to train and test columns demonstrate the similarity between the generated identities to the training or testing data, respectively, in terms of distribution.

As shown in the table, the random method obtains the highest values for all measures, as expected, except for the “SWD to test” result. The NN method is constrained only to geometries previously appearing in the data, and therefore demonstrates an identity distribution closely resembling the training data. This, however, prohibits the formation of high quality reconstructions, as well as the generalization to new identities. In addition, the GAN method produces more realistic geometries, as was also demonstrated by Table 2. Nevertheless, the identity distribution and geometry reconstruction were worse compared to other proposed methods. We suspect that, since the geometries produced by the GAN are derived from a model that is richer than the 3DMM, more training samples are required to learn the relationship between the textures and geometries by MAP or LS. As noted before, the MAP and LS methods produce similar results. However, the LS method is much easier to implement, and requires no assumptions on the data distribution. We conclude that, for our purpose, the LS method seems to be the most suitable among the proposed methods for fitting geometries to textures as well as for producing new identities.

10 DISCUSSION

A new model for generating high detail textures and corresponding geometries of human faces was introduced. The underlying idea was that an effective method for processing geometric surfaces via CNNs is to first align the geometric data-set to a template model achieving a universal parametrization to the class of surfaces we would like to process. This parametrization defines a map for each surface into a 2D image. Once in image form, a GAN loss can be employed to train a generator model that aims to imitate the distribution of the training data images. We further showed that by training a generator for both textures and geometries it is possible to synthesize high-detail textures and geometries that are linked to each other by a unique canonical mapping.

In addition, we introduced several methods for fitting 3DMM geometries by learning the underlying relation between texture and geometry, a relation that has been largely neglected in previous efforts. In Section 7.2, we also provide a quantitative and qualitative evaluation of each geometry reconstruction method. The proposed face generation pipeline therefore consists of a high resolution texture generator combined with a geometry that was either produced by a similar geometric generation model or by employing a learning scheme that produces the most likely corresponding 3DMM coefficients.

Besides the main pipeline, we introduced two additional data processing steps that improve sample quality. In Section 5.3, we described the design and construction of our canonical mapping. Our
mapping by design is intended to reduce distortion in important high detail areas while spreading the flattening distortion to non-essential areas. Our mapping was also designed to take maximal advantage of the available area in each image. In Section 5.3, we also showed that the suggested improved mapping, compared to Reference [31], indeed preserves delicate texture details in our predefined high importance regions. In Section 6, we also present a new technique for dealing with partially corrupted data. This is especially important when the data acquisition is expensive and prone to errors. By adding a corruption mask to the data at train time, the network is able to ignore the affected areas while still learning from the mostly unaffected ones. In the case of a typical data-set, this increases the amount of usable data by roughly 20%.

To evaluate the proposed model, we performed an extensive quantitative as well as qualitative analysis of several aspects of our model. Our main objective was to create a realistic model, a requirement that we break down into several factors. The proposed model produces high quality plausible facial textures that look as much like the training data as possible, but also compose novel faces not seen during training rather than repeat previously seen faces. To that end, we use an efficient approximation of Wasserstein distance between distributions to evaluate the local and global features of the produced textures and geometries as well as the distance between distributions of real and generated identities. Our results show that in both identity distribution and image feature resemblance the proposed model outperforms the 3DMM one, the most widely used model to date, and the only one that defines a process for generating new faces rather than reconstructing existing ones.

ACKNOWLEDGMENTS
We would like to thank Intel RealSense group for sharing their data and computational resources with us.

REFERENCES
[1] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. 2016. OpenFace: A General-purpose Face Recognition Library with Mobile Applications. Technical Report. CMU-CS-16-118, CMU School of Computer Science.
[2] Anil Bas, Patrik Huber, William A. P. Smith, Muhammad Awais, and Josef Kittler. 2017. 3D morphable models as spatial transformer networks. In Proceedings of the ICCV Workshop on Geometry Meets Deep Learning. 904–912.
[3] Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In Proceedings of the 26th Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., 187–194.
[4] Blender Online Community. 2017. Blender—A 3D Modelling and Rendering Package. Blender Foundation, Blender Institute, Amsterdam. Retrieved from: http://www.blender.org.
[5] James Booth, Epameinondas Antonakos, Stylianos Ploumpis, George Trigeorgis, Yannis Panagakis, Stefanos Zafeiriou et al. 2017. 3D face morphable models “in-the-wild.” In Proceedings of the IEEE Conference on ComputerVision and Pattern Recognition.
[6] James Booth, Anastasios Roussos, Allan Ponniah, David Dunaway, and Stefanos Zafeiriou. 2018. Large scale 3D morphable models. Int. J. Comput. Vis. 126, 2–4 (2018), 233–254.
[7] James Booth, Anastasios Roussos, Stefanos Zafeiriou, Allan Ponniah, and David Dunaway. 2016. A 3D morphable model learnt from 10,000 faces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 5543–5552.
[8] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: Going beyond Euclidean data. IEEE Sig. Proc. Mag. 34, 4 (2017), 18–42.
[9] Baptiste Chu, Sami Romdhani, and Liming Chen. 2014. 3D-aided face recognition robust to expression and pose variations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 1899–1906.
[10] Jiankang Deng, Shiyang Cheng, Niannan Xue, Yuxiang Zhou, and Stefanos Zafeiriou. 2018. UV-GAN: Adversarial facial UV map completion for pose-invariant face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’18).
[11] Michael S. Floater. 1997. Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geo. Des.* 14, 3 (1997), 231–250.

[12] Baris Gecer, Binod Bhattachari, Josef Kittler, and Tae-Kyun Kim. 2018. Semi-supervised adversarial learning to generate photorealistic face images of new identities from 3D morphable model. In *Proceedings of the European Conference on Computer Vision (ECCV’18).*

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Conference on Advances in Neural Information Processing Systems.* 2672–2680.

[14] Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika* 28, 3/4 (1936), 321–377.

[15] Ron Slossberg, Gil Shamai, and Ron Kimmel. 2018. Efficient inter-geodesic distance computation and fast classifi- cation.

[16] Ian T. Jolliffe. 1986. Principal component analysis and factor analysis. In *Principal Component Analysis.* Springer, 115–128.

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of GANs for improved quality, stability, and variation. In *Proceedings of the International Conference on Learning Representations (ICLR’17).*

[18] Davis E. King. 2009. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.* 10 (2009), 1755–1758.

[19] Or Litany, Alex Bronstein, Michael Bronstein, and Amesheh Makadia. 2018. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’18).*

[20] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, Nov. (2008), 2579–2605.

[21] Iacopo Masi, Anh Tuan Tran, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni. 2016. Do we really need to collect millions of faces for effective face recognition? In *Proceedings of the European Conference on Computer Vision.* Springer, 579–596.

[22] Julien Rabin, Gabriel Peyrè, Julie Delon, and Marc Bernot. 2011. Wasserstein barycenter and its application to texture mixing. In *Proceedings of the International Conference on Scale Space and Variational Methods in Computer Vision.* Springer, 435–446.

[23] 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.* Springer, 725–741.

[24] Elad Richardson, Matan Sela, and Ron Kimmel. 2016. 3D face reconstruction by learning from synthetic data. In *Proceedings of the 4th International Conference on 3D Vision (3DV’16).* IEEE, 460–469.

[25] Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. 2017. Learning detailed face reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’17).* IEEE, 5553–5562.

[26] Epameinondas Antonakos, Patrick Snape, Stefanos Zafeiriou, Iasonas Kokkinos, Riza Alp Güler, George Trigeorgis. 2016. DenseReg: Fully convolutional dense shape regression in-the-wild. Retrieved from: Arxiv:1612.01202 (2016).

[27] Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. 2017. Photorealistic facial texture inference using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’17).* Vol. 3.

[28] Matan Sela, Elad Richardson, and Ron Kimmel. 2017. Unrestricted facial geometry reconstruction using image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV’17).* IEEE, 1585–1594.

[29] Gil Shamai, Michael Zibulevsky, and Ron Kimmel. 2018. Efficient inter-geodesic distance computation and fast classi- fication scaling. *IEEE Trans. Pattern Anal. Mach. Intell.* (2018). https://ieeexplore.ieee.org/document/8509134/media#media.

[30] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. 2017. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’17).* Vol. 3. 6.

[31] Ron Slossberg, Gil Shamai, and Ron Kimmel. 2018. High quality facial surface and texture synthesis via generative adversarial networks. Retrieved from: Arxiv:Preprint Arxiv:1808.08281 (2018).

[32] Ayush Tewari, Michael Zollhöfer, Pablo Garrido, Florian Bernard, Hyeongwoo Kim, Patrick Pérez, and Christian Theobalt. 2018. Self-supervised multi-level face model learning for monocular reconstruction at over 250Hz. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’18).*

[33] Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gérard Medioni. 2017. Regressing robust and discriminative 3D morphable models with a very deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR’17).* IEEE, 1493–1502.
[34] Luan Tran and Xiaoming Liu. 2018. Nonlinear 3D face morphable model. Retrieved from: Arxiv Preprint Arxiv:1804.03786 (2018).

[35] Luan Tran, Xi Yin, and Xiaoming Liu. 2017. Disentangled representation learning GAN for pose-invariant face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 1415–1424.

[36] Thibaut Weise, Hao Li, Luc Van Gool, and Mark Pauly. 2009. Face/off: Live facial puppetry. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. ACM, 7–16.

[37] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In Proceedings of the Conference on Advances in Neural Information Processing Systems. 82–90.

Received November 2018; revised March 2019; accepted May 2019