Data-Driven Generative Models for the Deformation of Industrial Geometries

Guglielmo Padula

January 6, 2023

1 Preface

Our line of work consists in studying **generative models** for shape optimization of complex geometries with a large number of parameters; the objective is also to reduce the number of relevant geometrical parameters, for example for modeling naval hulls, and creating new artificial geometries similar to real data, as there are non-generative techniques for creating new real geometries (for example Free Form Deformation) but using them can be costly. Our data is artificially created data using divergence-free deformations on a real bulbus model provided by Fincantieri. We study mainly two class of models: Variational Autoencoders and Generative Adversarial Networks. With at a least one model for every type we arrive at a relative reconstruction error of 0.002.

2 Introduction

In the last five years there has been an increasing interest in using Deep Neural Network based models for generative new samples of 3D objects. There are also been some interest in deformation of 3d objects with the preservation of some characteristics, for example volume howewer we are not aware of the usage of generative models in this framework. The advantage of volume preserviness is important for industry because it could me same quantity of material used. We try to merge this gap by creating some generative models able to do deformations and preserve volume at the same time. Advantages of this methods is that typical methods can be very costly (see for example volume preserving Free Form Deformation. We test two main classes of models: Autoencoders and Generative Adversarial Networks. Another advantages of generative models is that the number of relevant parameter is greatly reduced.

3 Related Work

3.1 Volume preserving deformations

The first paper that talked of volume preserving deformations was Hirota et al [1] which developed a method for doing a variation of Free Form Deformation that preserved the volume of the deformed mesh with respect to the original.
This argument has furter been developed by Hahmann [2]. Von funck [3] et. all (2006) proposed a method for creating deformations using divergence free vector fields and so preserving the volume.
Eisemberg et al. [4] extends this method for shape interpolation, using the eigenvectors of the Von Neumann-Laplace equation for generating deformations.

3.2 Generative models for 3D meshes

In the last years there have been an increasing number of generative models for 3D mesh deformation.
Qtan [5] develops a Variational Autoencoder in which input data is encoded in the RIMD representation, which is rotation invariant.
Ranjan[6] develops a Convolutional Mesh Autoencoder (CoMA) which is based on Chebyscev Spectral Convolution.

Rana Hanocka [7] develops some new original convolutional layers for mesh data, and it also proposes an autoencoder.
Yuan [8] extend the work of both Hanocka and Ranjan.
Hahner [9] develop an Autoencoder model for semiregular meshes with different sizes.
Cheng [10] et. al develop a GAN model based on Berthelot (2017).

4 General metodology: CVPFFD and the Custom Layers

4.1 Assumption

The main assumption of all the paper is the the points of the mesh that we are going to deform are all in $K = \{(x, y, z) | x \geq 0, y \geq 0, z \geq 0\}$ and will belong to this set after the deformation, and that no triangles interior intersect ∂K. We we will apply the methods presented here to our mesh, we will restrict K to $(0, 1) \times [0, 1] \times (0, 1)$.

4.2 Constrained Volume Preserving Free Form Deformation

Let $D = [0, 1] \times [0, 1] \times [0, 1]$, n a scalar and $P_{ijk} = \left[\frac{i}{n}, \frac{j}{n}, \frac{k}{n}\right]$ (called control points), and let b_n^i the basis of $n + 1$ bernstein polynomials

$$b_{\nu,n}(x) = \begin{pmatrix} n \\ \nu \end{pmatrix} x^\nu (1 - x)^{n-\nu}$$

Then by the property of bernstein polynomials

$$\sum_{i=0}^{n} b_i^n(u) b_j^n(v) b_k^n(w) P_{ijk} = [u, v, w]$$

. So it is possible to define a map parametrized by $Q \in \mathbb{R}^{3(n+1)}$

$$T_Q(u, v, w) = \sum_{i=0}^{n} b_i^n(u) b_j^n(v) b_k^n(w)(P_{ijk} + Q_{ijk})$$

This map is continous and it is called Free Form Deformation map. If in general our mesh domain is K we can use a continous map $\phi : \mathbb{R}^3 \to \mathbb{R}^3$ such that $\phi(K) = \bar{K} \subseteq D$ and do $\phi^{-1} \circ T_Q \circ \phi$ to deform our space. So a Free Form definition is parametrized by ϕ and Q. In the literature K in surrounded by a bounding box and ϕ is simple a linear map. One of the main properties of Free Form Deformation is the possibility to choose where to deform the mesh by moving the appropriate control points.

Now a reasonable general definition of volume preserving free form deformation is a triple ϕ,Q, and $f : \mathbb{R}^{3(n+1)} \to \mathbb{R}^{3(n+1)}$ such that

$$Vol \circ \phi^{-1} \circ T_{f(Q)} \circ \phi(X) = Vol(X)$$

where X is our mesh. The problem of finding an FFD reduces to find an appropriate f that satisfies the constraint above. Notice that:

- given Q and ϕ more than one f satisfies the constraint.
- Not all the f are desiderable, for example the map $f(Q) = 0$ reduces the entire FFD to identity, an so preserves volume trivially.

To find an appropriate f let's start by finding an appropriate formula for the volume.

Theorem 1 *Let M a closed connected mesh with points $P = (x, y, z)$ and triangles T parametrized by triple of indices ordered such that all the face normals point outwards the mesh and that $x \geq 0, y \geq 0, z \geq 0$, it holds*

$$Vol(M) = \sum_{t \in T} \frac{(x_{t_1} + x_{t_2} + x_{t_3})}{6} \begin{vmatrix} (y_{t_2} - y_{t_1}) & (z_{t_2} - z_{t_1}) \\ (y_{t_3} - y_{t_1}) & (z_{t_3} - z_{t_1}) \end{vmatrix} =$$

$$\sum_{t\in T} \frac{(y_{t_1}+y_{t_2}+y_{t_3})}{6} \begin{vmatrix} (x_{t_1}-x_{t_2}) & (z_{t_1}-z_{t_2}) \\ (x_{t_3}-x_{t_2}) & (z_{z_3}-z_{t_2}) \end{vmatrix} =$$

$$\sum_{t\in T} \frac{(z_{t_1}+z_{t_2}+z_{t_3})}{6} \begin{vmatrix} (x_{t_1}-x_{t_3}) & (y_{t_1}-y_{t_3}) \\ (x_{t_2}-x_{t_3}) & (y_{z_2}-y_{t_3}) \end{vmatrix}$$

From here we see that the volume is trilinear in the point coordinates, so it is possible to act in a linear way on one coordinate at a time to obtain volume preservation. This is exactly the approach of [2], which we will briefly report here. The following theorem is the result of three subsequent optimization problems, all with unique solution. P^x_{ijk} will indicate the x-coordinate of P_{ijk} similar for y and z.

PROOF See [2]

VPFFD has an exact formula, however it may not safisfy our general assumptions. So we will consider only a subset of all possible VFFD.

Definition 1 (Constrained Volume Preserving Free Form Deformation)
If a VFFD respects

- $T_L(\bar{K}) >= 0$
- $T_L(\bar{K} \cap \{(x,y,z)|z=0\}) \subseteq \{(x,y,z)|z=0\}$
- $T_L(\bar{K} \cap \{(x,y,z)|y=0\}) \subseteq \{(x,y,z)|y=0\}$
- $T_L(\bar{K} \cap \{(x,y,z)|x=0\}) \subseteq \{(x,y,z)|x=0\}$
- $T_L(\bar{K} \cap \{(x,y,z)|x*y*z\neq 0\}) \subseteq \{(x,y,z)|x*y*z\neq 0\}$

then it will be called Constrained Volume Preserving Free Form Deformation.

By looking at the optimization problem in [2] it can be easily shown that

Theorem 2
If

$$Q \in \mathcal{S} = \{Q^x_{0jk}=0, Q^y_{i0k}=0, Q^z_{ik0}=0\} \cap \{(\frac{i}{l}+Q^x_{ijk})^2 \geq \sum_{i=1}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}(Q^x_{ijk})^2 \forall i=1...l\}$$

$$\cap\{(\frac{j}{m}+Q^y_{ijk})^2 \geq \sum_{i=0}^{l}\sum_{j=1}^{m}\sum_{k=0}^{n}(Q^y_{ijk})^2 \forall j=1...m\} \cap \{(\frac{k}{n}+Q^z_{ijk})^2 \geq \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=1}^{n}(Q^z_{ijk})^2 \forall k=1...n\}$$

then we have a CVFFD.

Theorem 3 *The set $\mathcal{S}$ is convex.*

PROOF $\mathcal{S}$ is the intersection of hyperplanes and volume internal to paraboloids, which are convex sets.

We remark that this is only a sufficient condition, empirically we saw that if we deform only the control points which are not on the boundary by small values then we still get a CVFDD.

4.3 Custom Layers

In all our generative models, we use two different custom layer in the sampling phase:

- A smoothing layer
- A volume layer

4.3.1 Smoothing Layer

This is a layer that applies laplacian smoothing

$$K_i = \frac{1}{|N_i|} \sum_{j \in N_i} K_j$$

where N_i are the neighbours of index i. Note that while K_i. Note that the laplacian smoother is applied only to points that are not on the boundary. Howewer the points of the boundary are still neighbours, so they influence the layer even if they are constant.

4.3.2 Volume Layer

If we modify Q^x by δ^x by theorem 1 we get that volume deviation is

$$\delta Vol(M) = \sum_{t \in T} \frac{(\delta^x_{t_1} + \delta^x_{t_2} + \delta^x_{t_3})}{6} \begin{vmatrix} (y_{t_2} - y_{t_1}) & (z_{t_2} - z_{t_1}) \\ (y_{t_3} - y_{t_1}) & (z_{t_3} - z_{t_1}) \end{vmatrix} = \sum f_i(M)^x \delta^x_i$$

where

$$f_i(M)^x = \sum_{T|t_1==i} - \begin{vmatrix} (y_{t_2} - y_{t_1}) & (z_{t_2} - z_{t_1}) \\ (y_{t_3} - y_{t_1}) & (z_{t_3} - z_{t_1}) \end{vmatrix} + \sum_{T|t_2==i} \begin{vmatrix} (y_{t_2} - y_{t_1}) & (z_{t_2} - z_{t_1}) \\ (y_{t_3} - y_{t_1}) & (z_{t_3} - z_{t_1}) \end{vmatrix} + \sum_{T|t_3==i} \begin{vmatrix} (y_{t_2} - y_{t_1}) & (z_{t_2} - z_{t_1}) \\ (y_{t_3} - y_{t_1}) & (z_{t_3} - z_{t_1}) \end{vmatrix}.$$

Same calculations holds for y and z. With this in mind we can write the following theorems

Theorem 4 *For every closed mesh M $f_i(M)$ has a variable sign.*

Proof Volume in invariant under translation so it must hold $\sum f^x_i = 0$.

Theorem 5 *The optimization problem*

$$\min_{\delta^x} \sum_i (\delta^x_i)^2$$

s.t

$$\begin{cases} \sum f^x_i \delta^x_i = a \\ \delta^x_i \geq -x_i + \epsilon \end{cases}$$

, has an unique solution.

Proof The problem is always feasible because the f_i have no constant sign. The objective function is stricly convex and the domain is convex, so the result is trivial.

Theorem 6 (Volume normalization layer) *Let K and G two meshes with all non-negative points. Let $a = \frac{1}{3}(Vol(K) - Vol(G))$ Let δ^x the solution of*

$$\min_{\delta^x} \sum_i (\delta^x_i)^2$$

s.t

$$\begin{cases} \sum f_i(G)^x \delta^x_i = a \\ \delta^x_i >\geq K^x_i + \epsilon \end{cases}$$

.

Let H such that $\begin{cases} H^x_i = G^x_i + \delta^x_i \\ H^y_i = G^y_i \\ H^z_i = G^z_i \end{cases}$

Let δ^y the solution of

$$\min_{\delta^y} \sum_i (\delta^y_i)^2$$

s.t

$$\begin{cases} \sum f_i(H)^y \delta_i^y = a \\ \delta_i^y >\geq K_i^y + \epsilon \end{cases}$$

Let I *such that* $\begin{cases} I_i^x = H_i^x \\ I_i^y = H_i^y + \delta_i^y \\ I_i^z = H_i^z \end{cases}$

Let δ^z *the solution of*

$$\min_{\delta^y} \sum_i (\delta_i^y)^2$$

s.t

$$\begin{cases} \sum f_i(I)^y \delta_i^y = a \\ \delta_i^z >\geq K_i^z + \epsilon \end{cases}$$

Let L *such that* $\begin{cases} L_i^x = I_i^x \\ L_i^y = I_i^y \\ L_i^z = I_i^z + \delta_i^z \end{cases}$

Then $Vol(I) = Vol(K)$.

Proof We already proved in theorem 5 that all the problems are feasibile and that they have an unique solution. Now by construction $Vol(L) = Vol(K) + 3 * \frac{1}{3}(Vol(K) - Vol(L)) = Vol(K)$.

5 The Data

We use a dataset which obtained by deforming a naval hull of Fincantieri usin Constrained Volume Preserving Free Form Deformation. We apply CVPFFD to deform only the bulb of a naval hull, leaving the rest unchanged: the boundary of the hull is composed by the set $\{x * y * z = 0\}$. The additional constraint guarantee that there will not be collision between points of the bulb and the other points. To avoid boundary problems we also deform only the points which are not on the boundary.

6 Generative models

The objective is to sample new 3d objects with quality comparable to the original ones. The new samples should also have the same volume. For this reason an explicit volume normalization layer is used. The generative models also use a pretrained PCA to increase stability and quality. We also use an explicit smoothing layer to increase quality. In the experiments reported for the latent space size we use the number of basis elements we used for the deformations (3), howewer also smaller sizes can be used, and indeed should be used if the number of basis elements is high (> 5) because of the curse of dimensionality. For estimating the quality of the generated 3d objects we decided to use Reconstruction Error, note that this is resonable because the divergence-free deformation basis may potentially generate all possible divergence-free deformations and so our dataset represents all the space. In all the models we decided to use a standard normal distribution for sample from the latent space.
All the models are trained with AdamW with gradient clipping. Every hidden units of our models are composed of:

- A linear layer
- A Normalization Layer (typically Batch Normalization)
- An Activation Layer (tipically ReLU)
- A Dropout Layer

General notations:

- $\mathbb{R}^{\mathbb{N}}$ latent space with a distance d_N
- X Hilbert space of the Data with associated distance d
- $Z \sim MultivariateNormal(0_N, I_N)$
- $A_\theta : X \to \mathbb{R}^N$ injective depending on a paramenter θ
- $B_\phi : \mathbb{R}^N \to X$ depending on a paramenter ϕ
- $C_\psi : X \to \mathbb{R}$ depending on a parameter ψ
- $D_\lambda : \mathbb{R}^N \to \mathbb{R}$
- $g : \mathbb{R} \to \mathbb{R}$ increasing
- $L_Y(y)$ will be likehood function of a random variable Y.
- $\mu_\theta = E[A_\theta(X)]$
- $\sigma_\theta = Var[(A_\theta(X))]^{\frac{1}{2}}$
- $\hat{Z}_\theta \sim MultivariateNormal(\mu_\theta, \sigma_\theta)$
- $f : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{N}}$ parametrized by β
- $\hat{\xi}_{X,n}^{Distribution}$ the MLE parameters associated to a distribution

6.1 Simple Autoencoder toy model

As a toy model for pure benchmarking we first implemented a very simple Autoencoder [11], composed of two parts: an Encoder which encodes the mesh in a latent space of dimension, and a Decoder that takes a point of the latent space and returns it to the data space. They are training together in such a way that the are one the inverse of the other. So with our notation:

$$\theta^\star, \phi^\star = \arg\min_{\theta,\psi} d(X, B_\phi(A_\theta(X)))$$

6.2 Simple WGAN toy model

A Generative Adversarial Network [12] is a generative model composed of a Generator and a Discriminator. The Generator tries to generate fake data and fooling the Discriminator and the Discriminator tries to discriminate between real and fake data. So with our notation:

$$\psi = \arg\max_{\psi} [g(C_\psi(X)) - g(C_\psi(B_\phi(Z_p)))]$$

$$\phi = \arg\max_{\phi} g(C_\psi(B_\phi(Z_p)))$$

Note that this is a minmax game and so it is general hard to train. For this reason we use a linear Discriminator as in the famous Wasserstain GAN paper [13].

6.3 (simplified) Variational Autoencoder model

A Variational Autoencoder [14] is the Probabilistic version of the AutoEncoder. The loss is usually approximated with the ELBO and the distribution are usually normal. Howewer as sampling with usual distribution does not guarantee volume preservation we use a deterministic decoder while keeping the probabilistic encoder. So the reconstruction part of the loss is substitued with the our d distance. We also use an hyperparameter α to improve the training (both of these techniques are also used in [5]).

$$\theta^\star, \phi^\star, \beta^\star = \arg\min_{\theta,\phi,\beta} [(1-\alpha) * d(X, B_\phi(A_\theta(X))) + \alpha * KL(\hat{Z}_\theta, Z)]$$

6.4 (simplified) Adversarial Autoencoder model

An Adversarial Autoencoder [15] is similar to a VAE, it is regularized with an adversarial newtwork. We adopt network with the deterministic encoder. We use the same simplification adopted in the VAE. In our notation:

$$\alpha \in (0,1)$$

$$\theta^\star, \phi^\star = \arg\max_{\theta,\phi} \alpha * (-d(X, B_\phi(A_\theta(X))) + (1-\alpha) * (E[g(C_\psi(\hat{Z}_\theta))])$$

$$\psi^\star = \arg\max_{\lambda} [E[g(D_\lambda(Z))] - E[g(D_\lambda(\hat{Z}_\theta))]]$$

6.5 BEGAN model

A Boundary Equilibrium GAN [16] is a GAN in which the Discriminator is an AutoEncoder. A simplified representation of its loss is:

$$\theta^\star, \phi_1^\star = \arg\min_{\theta,\phi_1} [d(X, B_{\phi_1}(A_\theta(X))) - d(X, B_{\phi_2}(\hat{Z}_\theta))]$$

$$\phi_2^\star = \arg\min d(X, B_{\phi_2}(Z))$$

6.6 VAEWGAN model

A VAEGAN [17] is similiar to an AAE model, howewer the discrinator acts on the space X, not in the latent space.

$$\gamma \in (0,1)$$

$$\mu_{\theta,\phi,\psi} = E[C_\psi(B_\phi(\hat{Z}_\theta))]$$

$$Y_{\theta,\phi,\psi} \sim MultivariateNormal(\mu_{\theta,\phi,\psi}, I)$$

$$\theta^\star = \arg\min_{\theta} KL(\hat{Z}_\theta, Z) - L_{Y_{\theta,\phi,\psi}}(C(X))$$

$$\phi^\star = \arg\min_{\phi} -\gamma * L_{Y_{\theta,\phi,\psi}}(C(X)) + g(C_\psi(\hat{X}_{\theta,\phi})) + g(C_\psi(B_\psi(Z))) - g(C_\psi(X))$$

$$\psi^\star = \arg\min_{\psi} -g(C_\psi(\hat{X}_{\theta,\phi})) - g(C_\psi(B_\psi(Z))) + g(C_\psi(X))$$

We implement a VAEGAN with some adjustment taken from the WGAN paper[13] (linear discriminator), this obtaining a VAEWGAN.

6.7 Evaluation

We use several metrics for evaluate our models.
Non geometric metrics:

- The reconstruction error
- The variance of the generated meshes

Then we check some geometric properties of the real and the generated meshes using the relative MMD distance which

$$RelMMD(F) = \frac{\sqrt{E[||\xi(F(X)) - \xi(F(B_\phi(Z)))||^2]}}{\sqrt{E[||\xi(F(X))||^2]} + \sqrt{E[||\xi(F(Y))||^2]}}$$

where xi is the basis function of the RKHS associated to the laplacian kernel and F is the property that we want to compare. We measure the distance of:

- Gaussian discrete curvature
- Total discrete curvature
- Area

We also compare results with the ones of some meshes with more strong deformation (with negative curvare in some points) to check the effectivennes of our measures.

6.8 Results

6.8.1 Hull

Note that the variance of the data is 0.064.

	AE	AAE	VAE	BEGAN	VAEWGAN	WGAN
$RelMMD(TC)$	0.60	0.57	0.60	0.558	a	a
$RelMMD(Area)$	0.03	0.01	0.003	0.016	a	a
$E[\min_X \frac{\|\|B_\phi(Z)-X\|\|}{\|\|X\|\|}]$	0.021	0.015	0.012	0.019	a	a
$Var(B_\phi(Z))$	0.026	0.03	0.024	0.068	a	a
$RelMMD(GC)$	0.189	0.10	0.114	0.14	a	a
$RelMMD(Id)$	0.013	0.0079	0.008	0.01	a	a

Colomns in green indicate models that have reached convergence in the training phase. Red columns indicate problems that have not reached convergence while the blue coloumn represents a dataset from which we don't want to sample from because it has non phisically admissible configurations (negative gaussian curvature in some points). The best model is the Adversarial Autoencoder, which has a slightly lesser variance then the other models, howewer it approximates best the distribution of the geometric properties.

References

[1] Gentaro Hirota, Renee Maheshewari, and Ming C. Lin. Meshgan: Non-linear 3d morphable models of faces, 1999.

[2] Stefanie Hahmann, Georges-Pierre Bonneau, Sébastien Barbier, Gershon Elber, and Hans Hagen. Volume-preserving FFD for programmable graphics hardware. 2011. DOI: 10.1007/s00371-011-0608-5.

[3] Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations, 2006.

[4] Marvin Eisenberger, Zorah Lähner, and Daniel Cremers. Divergence-free shape interpolation and correspondence, 2018. arXiv:1806.10417.

[5] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models, 2017. arXiv:1709.04307.

[6] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. Generating 3d faces using convolutional mesh autoencoders. 2018. arXiv:1807.10267.

[7] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. 2018. arXiv:1809.05910.

[8] Yu-Jie Yuan, Yu-Kun Lai, Jie Yang, Hongbo Fu, and Lin Gao. Mesh variational autoencoders with edge contraction pooling, 2019. arXiv:1908.02507.

[9] Sara Hahner and Jochen Garcke. Mesh convolutional autoencoder for semi-regular meshes of different sizes, 2022.

[10] Shiyang Cheng, Michael Bronstein, Yuxiang Zhou, Irene Kotsia, Maja Pantic, and Stefanos Zafeiriou. Meshgan: Non-linear 3d morphable models of faces, 2019. arXiv:1903.10384.

[11] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2020. arXiv:2003.05991.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. arXiv:1406.2661.

[13] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. arXiv:1701.07875.

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. arXiv:1312.6114.

[15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2015. arXiv:1511.05644.

[16] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks, 2017. arXiv:1703.10717.

[17] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric, 2015. arXiv:1512.09300.