Deep Geometric Texture Synthesis

farhad javid (1007541359)

December 2020

1 Introduction

Here, I present my final project for "Geometric Processing" course which is an implementation of the "Deep Geometric Texture Synthesis"[1]. The paper aims at learning an arbitrary texture of a mesh and generating a similar texture on another mesh. The technique requires; 1) a hierarchy of the coarse to fine meshes of the reference textured mesh for training, 2) a fully convolutional generative adversarial network that can learn the texture from the hierarchical training , i.e. coarse to fine, mesh, and 3) a training model on a single mesh that uses Wasserstein generative adversial training with geometric penalty and an extra reconstruction loss.

The convolutional model is made of fully convolutional layers with instance normalization and leaky ReLU. Since no polling is added the model is only capable of finding local features, e.g. features of the small patches, e.g. single faces or faces with their first-ring neighbors. A hierarchy of the training meshes from coarse to fine are then required to capture the full topological features of the textures.

2 Hierarchical Training Set

To prepare the training meshes, we start from a coarse mesh of the same topology of the reference texture mesh. The coarse mesh is then subdivided iteratively and morphed into the reference mesh in an optimization process. In each subdivision, the faces of the coarse mesh are divided to four faces but placing three vertices in the middle of the face' edges. The morphing is performed in an optimization process which minimizes the Chamfer distance between the subdivide mesh and the reference textured mesh by picking randomly distributed sample points on the training and the reference meshes. In addition to minimizing the Chamfer distance, the optimizer maximizes the cosine of the angle between the normals to the sampled points. Also, two requlariza tion terms are added to the optimization, one minimizes the variance of the edge lengths in the training mesh during morphing and the other minimize the distance between each vertex and the average of its one-ring neighbors. I have not modeled this optimization completely. I have only considered the Chamfer distance in my

modeling. My code gets a coarse template mesh and a fine target mesh and based on a backpropagation technique tries to conform the training mesh to the target one.

3 Convolutional Model

The generative adversarial networks (GAN) is a fully convolutional model made of seven layers of 2D convolutions with *instance normalization* and *leaky ReLU*. The convolutions are performed over the mesh faces. The first convolutional layer is a single face convolution while the following ones include the 1-ring of each face. No pooling layer added because pooling requires to collapse the mesh faces which affect the textures geometry. The pooling however replaced by the hierarchical training of the system on a series of coarse to fine meshes.

4 Training

The training of the GAN model is performed on a single texture mesh which is generated on a hierarchy of coarse to fine meshes. The coarser meshes train the system for large texture features while the finer one capture the fine texture geometric features. A Wasserstein loss with a gradient penalty is used to train the system.

5 My implementation

5.1 Generating Training Set

I have implemented a simplified version of generating the texture training set. The training starts from a coarse template mesh. This mesh is iteratively subdivided and morphed into a reference *textured* mesh. The morphing is performed by minimizing the Chamfer distance between the subdivided training mesh and the reference texture mesh. In addition to the Chamfer distance, the objective function includes a negative of the cosine between the normals to the training and the reference mesh. Two regularization terms, one for minimizing the edges variance and the other for minimizing the distance between each vertex and its 1-ring average are suggested which are not included in my implementation. I have used a backpropagation technique to minimize this objective function (please see `/deep_texture/src/deep_texture_training_data.cpp` for the optimization implementation and `/deep_texture/main.cpp` for how to use it). Unfortunately, my code is not capable of successfully morphing the coarse template meshes to the reference texture, as can be seen in the Fig.1.

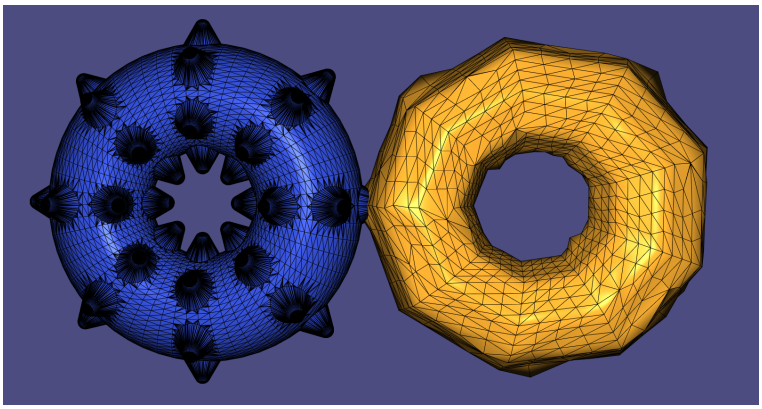

Figure 1: A sample of generated training texture mesh.

5.2 Convolutional Model and Training

I have also implemented the training model in `training.py` and also the convolutional model is partially implemented in `model.py`. However, since I do not have any training texture mesh, I could not test my model.

References

[1] Amir Herz, Rana Hanocka, Raja Giryesh, and Daciel Cohen-Or. *Deep geometric Texture Synthesis*. arXiv preprint arXiv:2007.00074, 2020.